LIÈGE université
Sciences Appliquées

Network Programming with SRv6

A thesis submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy (PhD) in Engineering Science

by

Clarence FILSFILS

Supervisor: Guy LEDUC

DOCTORAL COLLEGE IN ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

APRIL 2020

Jury

- Prof. Laurent Mathy, Université de Liège, President,
- Prof. Guy Leduc, Université de Liège, Supervisor,
- Prof. Benoit Donnet, Université de Liège, Belgium
- Prof. Serge Fdida, Sorbonne Université, Paris, France,
- Prof. Pascal Mérindol, Université de Strasbourg, France,
- Prof. Stefan Schmid, Universität Wien, Austria

Abstract

This PhD thesis defines the Network Programming for IPv6 also known as Segment Routing (SR) with IPv6 data plane (SRv6). SRv6 leverages the source routing paradigm by allowing a source to engineer a flow across a network as a program: i.e. a combination of traffic engineering objectives, service chainings, and Virtual Private Network (VPN) instantiations.

The network program is encoded in the Segment Routing (SR) extension Header (SRH) of a network packet as an ordered list of 128-bit segments. Each segment represents an instruction (topological, service, VPN). The first segment is placed in the destination address of the packet. The most significant bits of a segment are called “locator”. The locator acts as any routed subnet address and ensures that the packet destined for a segment is routed to the parent of that segment. Once at the parent the rest of the segment (called function) is mapped to a pseudocode enabling programmability, then the pseudocode is executed, the next segment in the SRH is placed in the destination address, and the packet is routed accordingly.

A function can be associated with any behavior: traffic engineering (e.g., take the shortest path to this node, take the shortest path to this node and then take this specific interface), service (e.g., a firewall application in a container), or VPN (e.g., look the updated destination address in this virtual forwarding table).

The network acts as a big computer. The packet goes from node to node and receives sequential processing according to ordered instructions selected by the source and encoded in the SRH. No intermediate node stores any a priori state for the flow. The only per-flow state is in the packet header.

In this thesis, we explain how SRv6 deployed within a Service Provider (SP) domain delivers the following benefits:

- Stateless-ness: Transit nodes must not store per-engineered flow state
- Scalability: The solution must support multi-domain SP networks with 100 thousands of routers
- Hardware-friendliness: Line rate performance without significant cost increase
- Explicit Routing (intra- and inter-domain): Ability to take a path different than the best-effort shortest-path delivered by IGP/BGP routing protocols
- Sub-50msec Prefix-Independent Protection against link/node/SRLG failure
- Micro-Loop Avoidance: Avoid transient loops during intra-domain routing protocol convergence
- Overlay Virtualization: Ability to create Virtual Private Networks (VPN)
- Service Chaining with Metadata: Ability to steer the engineered flow through a set of services (virtual or not) and pass meta-data between these services. The information may be used for monitoring, service chain modification etc.
- Optimum Load-Balancing: The load-balancing within the domain must leverage all the available flow entropy
- MTU Efficiency: The SR overhead must be minimized

We end the thesis with a report of the rich SRv6 ecosystem built in record time. By November 2019, we count 18 hardware implementations, 9 open source applications and 7 significant commercial deployments.

Résumé

Cette thèse de doctorat définit “la programmation réseau pour IPv6”, aussi appelée routage par segment avec un plan de données IPv6 (SRv6). SRv6 permet à une source de programmer le réseau afin d’obtenir un service spécifique pour un flux donné. Un service est une combinaison d’objectif d’ingénierie du trafic (p. ex. un chemin à faible délai), d’un chaînage de services (p. ex. passer un pare-feu lors de la connexion à Internet) et d’une instanciation de VPN (à la sortie du domaine SR, rechercher l’adresse de destination du paquet dans la table d’acheminement dédiée au client VPN).

Le programme SRv6 est codé dans l’extension SR de l’en-tête (appelé SRH) d’un paquet réseau sous la forme d’une liste ordonnée de segments de 128 bits. Chaque segment représente une instruction (topologique, service, VPN). Le premier segment est placé dans l’adresse de destination du paquet. Les bits les plus significatifs d’un segment sont appelés “Locator”. Le “locator” agit comme n’importe quel préfixe routé et s’assure que le paquet destiné à un segment arrive à l’origine de ce segment. Une fois l’origine du segment atteinte, le reste du segment (appelé fonction) est associé à un pseudo-code, le pseudo-code est exécuté, le segment suivant dans le SRH est placé dans l’adresse de destination et le paquet est acheminé en conséquence.

Le réseau agit comme un grand ordinateur. Le paquet passe de nœud en nœud pour recevoir un traitement séquentiel en fonction d’instructions ordonnées sélectionnées par la source et codées dans la SRH. Aucun nœud intermédiaire ne stocke d’état a priori pour le flux. Le seul état spécifique au flux est dans l’en-tête du paquet.

Dans cette thèse, nous expliquons comment SRv6 déployé dans un domaine de fournisseur de service (SP) offre les avantages suivants :

- Absence d’état : les nœuds de transit ne doivent pas stocker d’état pour le flux programmé
- Passage à l’échelle : la solution doit prendre en charge les réseaux SP multidomaines avec 100.000 routeurs.
- Efficacité hardware : performance au débit des liens physiques, sans augmentation significative de coût.
- Routage explicite (intra et inter domaine) : capacité de prendre un chemin différent du chemin le plus court choisi par les protocoles de routage IGP / BGP.
- Protection contre la perte de trafic en cas de panne soudaine d’un lien ou nœud.
- Evitement de boucles transitoires lors de la convergence du protocole de routage.
- Possibilité de créer un réseau privé virtuel (VPN).
- Chaînage de services avec métadonnée : possibilité de programmer un flux à travers un ensemble de services (virtuels ou non) et de transmettre des métadonnées entre ces services. Ces informations peuvent être utilisées pour la surveillance ou la modification de la chaîne de services.
- Equilibrage optimal de la charge en exploitant toute l’entropie de flux disponible.
- Efficacité MTU : la surcharge due au SRH doit être minimisée.

Nous terminons la thèse par un rapport sur le riche écosystème SRv6 construit en un temps record. En novembre 2019, nous comptons 18 implémentations matérielles, 9 applications open source et 7 déploiements commerciaux importants.

Acknowledgements

I would like to thank our lead operators: Rani Assaf, Dan Bernier, Alex Bogdanov, Dennis Cai, Gaurav Dawra, Bruno Decraene, Kentaro Ebisawa, Niels Hanke, Martin Horneffer, Bart Jenssen, Mazen Kaddham, Anton Karneliuk, Bart Janssen, Zhichun Jiang, John Leddy, Stephane Litkowski, Satoru Matsushima, Takuya Miyasaka, Paul Mattes, Sebastien Parisot, Robert Raszuk, Dirk Steinberg, Yukito Ueno, Jim Uttaro, Dan Voyer, Shawn Zandi. Their genuine motivation to drastically improve the IP technology was required to surpass the barriers of inertia along our journey.

I would like to thank our academic partners (Professors Olivier Bonaventure, Antonio Cianfrani, Stefano Salsano and Domenico Siracusa) who jumped with energy into the vast space of un-explored opportunities created by the new SR paradigm.

I would like to thank the Cisco executive team (Ravi Chandra, Jonathan Davidson, Vipul Deokar, David Goeckeler, Venu Venugopal, David Ward, Kevin Wollenweber) for their confidence and empowerment. Ideas only become great when built and deployed at scale. The significant investment in the SR project way before the industry recognized its merits shows vision, ability to take risk and innovate at scale.

I would like to thank Swadesh Agrawal, Zafar Ali, Jisu Bhattacharya, Bertrand Duvivier, Pablo Camarillo, Francois Clad, Darren Dukes, Rakesh Gandhi, Jakub Horn, Junaid Israr, Jos Liste, Kris Michielsen, Peter Psenak, Kamran Raza, Kiran Sasidharan, Ianik Semco, Siva Sivabalan, Ketan Talaukilar, Frederic Trate, Richard Valle, the entire Cisco SR team (Appendix 4). Only three words need to be said: “What a team!”.

I would like to thank Professor Andre Danthine for seeding this thesis many years ago. His problem statement always stayed in my mind along these years.

I would like to thank John Bettink, Garry Epps, Michael Laor, Bob Olson, Mo Tatar, David Tsiang and David Ward. This dream team built the 12000 and CRS product lines. I learned so much working close to them.

This thesis would not exist without the excellent advice from Professor Guy Leduc.

I would like to thank Valerie and my friends. They mean so much to me.

Note: naming seemed needed to pay the deserved tribute to each person. Doing so exposes me to the likely regret to have missed some names. To these persons, I present my sincere apology.

Table of Contents

List of Acronyms

A	
Adj-SID	Adjacency Segment Identifier
ARGS	Arguments
AS	Automated Steering
ATM	Asynchronous Transfer Mode
B	
BGP	Border Gateway Protocol
BGP-EPE	Border Gateway Protocol - Egress Peer Engineering
BGP-LS	Border Gateway Protocol Link-State
BSID	Binding Segment Identifier
C	
CRH	Compressed Routing Header
D	
DA	Destination Address
DCI	Data Center Interconnect
DPI	Deep Packet Inspection
E	
ECMP	Equal Cost Multi-Path
END	Endpoint
End.AD	Endpoint to IPv6 SR-unaware APP via dynamic proxy
End.AM	Endpoint to IPv6 SR-unaware APP via masquerading proxy
End.AS	Endpoint to IPv6 SR-unaware APP via static proxy
End.B6	Endpoint bound to an SRv6 policy
End.B6.Encaps	Endpoint bound to an SRv6 encapsulation Policy
End.DT4	Endpoint with decapsulation and IPv4 table lookup
End.DT6	Endpoint with decapsulation and IPv6 table lookup
End.DX2	Endpoint with decapsulation and Layer-2 cross-connect
End.DX4	Endpoint with decaps and IPv4 cross-connect
End.DX6	Endpoint with decapsulation and IPv6 cross-connect
End.T	Endpoint function with specific IPv6 table lookup
End.X	Endpoint with Layer-3 cross-connect
EVPN	Ethernet Virtual Private Network

F	
FC	Fast Convergence
FIB	Forwarding Information Base
Flex-Algo	Flexible Algorithm
FR	Frame Relay
FRR	Fast Re-route
FUNCT	Function
G	
GPRS	General Packet Radio Service
GTP	GPRS Tunneling Protocols
H	
Hdr Ext Len	Header Extension Length
I	
IACL	Inbound Infrastructure Access Control List
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IGP	Interior Gateway Protocol
Inter-AS	Inter Autonomous-System
IoT	Internet of Things
IP	Internet protocol
IPv6	Internet protocol version 6
IS-IS	Intermediate System to Intermediate System
L	
L2VPN	Layer 2 Virtual Private Network
L3VPN	Layer 3 Virtual Private Network
LDP	Label Distribution Protocol
LFA	Loop-Free Alternate
LISP	Locator ID Separation Protocol
LOC	Location
LSDB	Link-State Database
LSP	Label-Switched Path

M	
MPLS	Multiprotocol Label Switching
MPLS-TE	Multiprotocol Label Switching – Traffic Engineering
MTU	Maximum Transmission Unit
N	
NANOG	North American Network Operators' Group
NETCONF	Network Configuration Protocol
NFV	Network Function Virualization
NH	Next Header
NSH	Network Service Header
O	
OAM	Operations, Administration, and Maintenance
ODN	On-Demand Next Hop
OSPF	Open Shortest Path First
P	
P2MP	Point-to-MultiPoint
PCE	Path Computation Element
PCEP	Path Computation Element Communication Protocol
PW	Pseudo-Wire
Q	
QoS	Quality of Service
R	
RIB	Routing Information Base
RIPE	Réseaux IP Européens
RR	Route Reflector
RSVP	Resource Reservation Protocol
RSVP-TE	Resource Reservation Protocol – Traffic Engineering
S	
SDN	Software Defined Networking
SDWAN	Software-Defined Wide-Area Network
SERA	SEgment Routing Aware Firewall
SID	Segment Identifier
SL	Segments Left
SLA	Service Level Agreement

SP	Service Provider
SPRING	Source Packet Routing in Networking
SR	Segment Routing
SR-DB	Segment Routing Database
SR-MPLS	Segment Routing for Multiprotocol Label Switching dataplane
SR-TE	Segment Routing – Traffic Engineering
SRDM	Segment Routing Demand Matrix
SRH	Segment Routing Header
SRLG	Shared Risk Link Group
SRP	Segment Routing Policy
SRv6	Segment Routing for IPv6 dataplane
SWAN	Software-driven WAN
T	
T.Encaps	Transit behavior with encapsulation in an SRv6 policy
T.Encaps.L2	T.Encaps behavior of the received L2 frame
T.Encaps.Red	Transit with reduce encapsulation in an SRv6 Policy
T.Insert	Transit behavior with insertion of an SRv6 policy
TE	Traffic Engineering
TILFA	Topology Independent Loop-Free Alternate
TLV	Type-Length-Value
U	
UDP	User Datagram Protocol
uSID	Micro-SID
V	
VPN	Virtual Private Network
VRF	Virtual Routing and Forwarding
VxLAN	Virtual extensible Local Area Network

1 INTRODUCTION AND PROBLEM STATEMENT

1.1 Introduction

Historically, within its domain, a Service Provider (SP) has only used the IP layer to deliver the most basic best-effort routing service. Value-added services have been delivered via layers below (MPLS) and above IP (UDP/VxLAN, NSH).

The scope of this thesis is the search for an IP-native solution for the delivery of these services.

While clearly the elimination of now-redundant layers will deliver a much simpler solution to build and operate (e.g. the removal of the MPLS layer decreases the forwarding table hardware requirement by a factor close to 3), the point of our study is that by doing less, we can do better.

To this end we propose a solution called "SRv6 Network Programming", which allows the delivery of the value-added services without laying any state on transit routers. This is a major improvement as scale is the number one issue of SPs that need to deliver services for up to several hundred million human subscribers and billions of sensors.

1.2 Problem Statement

We seek an IP-based integrated solution that supports the following characteristics:

- Stateless-ness
  - Transit nodes must not store per-engineered flow state
- Scalable
  - The solution must support multi-domain SP networks with 100 thousand of routers or data-centers with billions of servers
- Hardware-friendly
  - Line rate performance without significant cost increase
- Explicit Routing (intra and inter-domain)
  - Ability to take a path different than the best-effort shortest-path delivered by IGP/BGP
- Sub-50msec Prefix-Independent Protection against link/node/SRLG failure
- Micro-Loop Avoidance
  - Avoid transient loops during IGP network convergence
- Overlay Virtualization
  - Ability to create Virtual Private Networks (VPN)
- Service Chaining with Metadata
  - Ability to steer the engineered flow through a set of services (virtual or not) and pass information between these services. The information may be used for monitoring, service chain modification etc.
- Optimum Load-Balancing
  - The load-balancing within the domain must leverage all the available flow entropy
- MTU Efficiency
  - The SR overhead must be minimized

We limit the scope of this document to the delivery of the above benefits within a Service Provider (SP) domain: i.e., for the transport of a customer packet from an ingress edge to an egress edge.

The domain may be composed of hundreds of sub-domains with a total of tens to hundreds of thousands of routers. The scale requirement is huge and is not limited by the SP domain scope.

The service is most likely provided as an independent overlay.

The service requires stringent underlay Service Level Agreements (SLA), which require explicit routing, sub-50msec prefix-independent protection against link/node/SRLG failure and micro-loop avoidance.

Several virtual or physical services (e.g., firewall) must be applied through the service. Metadata must be shareable between these services (e.g., user identity, credentials, performance monitoring, proof of service processing).

The sum of these services must not deteriorate load balancing within the highly meshed SP network.

The delivery of these services must not incur a significant header overhead.

The service must be secure.

2 STATE OF THE ART

2.1 Distributed hop-by-hop routing (IP)

The IP solution is based on hop-by-hop independent forwarding decisions. Network-wide loop-free forwarding is ensured by a distributed underlay routing protocol [1] [2]. Services routes (VPN, overlay) are distributed by BGP [3] and recurse on an IGP path to the BGP next-hop.

The base IP solution does not support explicit routing and hence cannot provide cost vs delay service differentiation or disjoint services.

The IP solution scales very well thanks to its inherent characteristics: distributed, stateless and overlay/underlay recursion.

2.2 Centralized Circuit Routing (MPLS RSVP-TE)

MPLS RSVP-TE [82] has been designed as a version of ATM for IP.

An RSVP-TE LSP is defined as a stateful circuit: each hop holds state and switches the received frame based on the incoming circuit ID. A distributed accounting system allocates the bandwidth between competing circuits. As it is distributed, this leads to race conditions, slow convergence, nondeterministic outcome and suboptimal resource allocation [4].

From a bandwidth admission control viewpoint, it seems safe to write that there has been weak deployment and that the few who deployed have reported complex operation models and scaling issues [5] [6].

In fact, most of the RSVP-TE deployments have been limited to the fast re-route (FRR) use-case.

Our point is not to criticize the RSVP-TE protocol definition or minimize its merits. 20 years ago, there were good reasons for defining RSVP-TE and MPLS TE the way they appear today. 20 years ago, RSVP-TE and MPLS-TE provided a major innovation to IP networks. At that time, there was no other bandwidth optimization solution or FRR solution. RSVP-TE and MPLS-TE introduced great benefits 20 years ago.

Our point is to look at RSVP-TE applicability in IP networks in 2020 and beyond. Does it fit the needs of modern IP networks?

In our opinion, RSVP-TE and the classic MPLS TE solution have been defined to replicate FR/ATM in IP. The objective was to create circuits whose state would be signaled hop-by-hop along the circuit path. Bandwidth would be booked hop-by-hop. Each hop's state would be updated. The available bandwidth of each link would be flooded throughout the domain using IGP to enable distributed TE computation.

We believe that these design goals are no longer consistent with the needs of modern IP networks.

First, RSVP-TE is not ECMP-friendly. This is a fundamental issue as the basic property of modern IP networks is to offer multiple paths from a source to a destination. This ECMP- nature is fundamental to spread traffic along multiple paths to add capacity as required and for redundancy reasons.

Second, to accurately book the used bandwidth, RSVP-TE requires all the IP traffic to run within so-called “RSVP-TE tunnels”. This leads to much complexity and lack of scale in practice.

Let us illustrate this by analyzing the most frequent status of a network: i.e. a correctly capacity-planned network.

Such a network has enough capacity to accommodate without congestion a likely traffic volume under a set of likely independent failures. The traffic is routed according to the IGP shortest-path and enough capacity is present along these shortest-paths. This is the norm for the majority of SP and Enterprise networks either all the times or at least most of the times (this is controlled by the “likeliness” of the traffic volume and the failure scenarios).

Clearly, in these conditions, traffic engineering to avoid congestion is not needed. It seems obvious to write it but as we will see further, this is not the case for an RSVP-TE network.

In the rare cases where the traffic is larger than expected or a failure occurs, congestion occurs, and a traffic engineering solution may be needed. We write “may” because once again it depends on the capacity planning process.

Some operators might capacity plan the network via modeling, so that these occurrences are so unlikely that the resulting congestion might be tolerated. This is a very frequent approach.

Some other operators may not tolerate even these rare congestions and then require a tactical traffic-engineering process.

A tactical traffic-engineering solution is a solution that is used only when needed.

To the contrary, the classic RSVP TE solution is an “always-on” solution. At any time (even when no congestion is occurring), all the traffic must be steered along circuits (RSVP-TE tunnels). This is required to correctly account the used bandwidth at any hop.

This is the reason for the infamous full-mesh of RSVP-TE tunnels. Full-mesh implies that there must be a tunnel from anywhere to anywhere on the network edge and that all traffic must ride on RSVP-TE tunnels. IP forwarding spoils accurate traffic statistics.

Hence, traffic can never utilize IGP derived ECMP paths and to hide the lack of ECMP in RSVP-TE, several tunnels must be created between each source and destination (at least one per ECMP path).

Hence, while no traffic engineering is required in the most likely situation of an IP network, the RSVP-TE solution always requires N^2*K tunnels where N scales with the number of nodes in the network and K with the number of ECMP paths. While no traffic engineering is required in the most likely situation of an IP network, the classical MPLS TE solution always requires all the IP traffic not to be switched as IP, but as MPLS TE circuits.

The consequence of this “full-mesh” is lots of operational complexity and limited scale, most of the time, without any gain. Indeed, most of the times, all these tunnels follow the IGP shortest-path as the network is correctly capacity planned and no traffic engineering is required.

This is largely suboptimal. An analogy would be that one needs to wear his raincoat and boots every day while it rains only a few days a year.

Let us remember the origins of the classical RSVP-TE complexity and scale problem:

- the use of a circuit as fundamental unit of traffic engineering
- the per-hop states associated with a circuit
- k * N^2 state scaling
- full-mesh operational complexity
- the non-ECMP nature of a circuit
- the distributed nature of the bandwidth accounting leading to race conditions (slow convergence and nondeterministic output) and suboptimal resource allocation

Finally, a key reason for the RSVP-TE complexity is the lack of realization that most networks are correctly planned and hence that traffic engineering is rarely needed.

"Wear a raincoat and boots every day while it rains only a few days a year"

This is one of the key intuitions leading to our proposed solution.

2.3 Source Routing

Source Routing allows a source to encode an end-to-end path as a series of waypoints. A path in-between waypoints may either be an interface or a distributed ECMP-aware shortest path.

Source Routing has been defined both for IPv4 and IPv6.

A salient point to note is that in the prior-art before segment routing, an IP waypoint has been defined as an IP address. In the "Network Programming Solution" we define a waypoint as a network instruction called a segment. The locator part of the segment is routed like an address. The function part of the segment enables programmability: any instruction, pseudo-code, container, virtual machine can be bound to the function. The function part of the segment and the "network programming solution" was not proposed in the IP source routed solutions.

IPv4 source routing has never been used commercially and was disabled by default [7].

IPv6 Type 0 Routing Header has been deprecated [8] with an explicit note that future extension headers may revisit the matter.

The deprecation reason (security in the Internet) is not relevant to the scope of this document (deployment within a controlled domain). Within a controlled domain, there are obvious and well-known security measures to prevent the issues reported in [8]. We explain these later in this document.

Another salient point to note is that the IPv4 and IPv6 source routing headers did not foresee the carriage of metadata. In our solution, we model a network program as a series of instructions in the packet header. Intuitively, like a computer, we need shared memory between these instructions to pass information as the program executes. The extension header we propose not only contains a list of segments, but it also contains a flexible TLV-based mechanism to transport the metadata required by the network program.

2.4 Overlay Virtualization

Overlay virtualization (also called Virtual Private Network, VPN) is typically provided by MPLS-VPN [9] or VxLAN [10].

In a nutshell, the ingress edge of the SP network, encapsulates the customer packet in an outer shell (MPLS stack or IP/UDP/VxLAN stack of headers), which is then transported through the network up to an egress edge node. The egress edge node uses some field in the outer shell to associate the inner packet with a private forwarding table (often called VRF). It decapsulates the inner packet and forwards it based on the associated table.

In many deployments, the overlay is decoupled from the underlay (e.g. VxLAN for the overlay, SR-MPLS for the underlay). This leads to inefficiencies such as more complex troubleshooting and MTU overhead.

2.5 Service Chaining

Service Chaining consists in constraining an IP flow through a sequence of services (physical or virtual).

The state of the art consists in using policy-based routing in cascaded back-to-back VPN constructions. This requires significant state at each service hop and hence complex orchestration.

The NSH [11] proposal inherently keeps these drawbacks: statefulness and hence complex orchestration.

2.6 IPv6 is happening

It suffices to read the following reference to understand the obviousness of the availability of a healthy 1.2 billion consumer IPv6 economy [12].

2.7 5G is happening

5G is happening and requires hyper scale, stateless, explicit routing (latency versus cost, plane differentiation), integrated service chaining with metadata [13].

5G both provides the motivation for the genuine benefits of the Network Programming solution but also provides the economical funding to make it happen.

2.8 Alternative solution

IP tried to solve these problems one by one by introducing independent shim layers below and above the IP layer:

- MPLS-RSVP-TE: stateful circuit-based routing below IP for TE
- UDP: used for its load-balancing property: above IP
- VxLAN: overlay virtualization: above IP

- NSH: stateful service chaining: above IP

Aside not solving these problems successfully, these independent solutions create new interworking problems (e.g. how does VxLAN encapsulation interwork with NSH encapsulation).

3 THOUGHT PROCESS AND PROPOSED SOLUTION

3.1 A seed in 1996

In 1996, Professor Danthine proposed a PhD thesis on "The evolution of the IP layer".

I felt that I did not understand enough the current IPv4 solution to be able to propose any valuable alternative and hence I decided to join Cisco Systems to better learn the state-of-the-art.

However, the seed was planted and later, I would always think: "why did they do it like this", "how would I do it, if I had the opportunity".

3.2 MPLS architecture and deployment

In 1998, I moved to the Cisco European consulting team to deploy a new technology called tag-switching (later renamed MPLS).

This was a fantastic experience: witnessing the entire technology definition process, working closely with the MPLS architecture team, having first-hand experience designing, deploying the first and biggest MPLS networks and collecting feedback and requirements from operators.

Over these years, while the elegance of the MPLS data plane has rarely been challenged, it became obvious that the MPLS "classic" (LDP and RSVP-TE) control-plane was too complex and lacked scalability.

3.3 QoS deployments

From 1998 to 2005, I lead the first large-scale QoS deployments worldwide and helped engineer the software and hardware QoS functionality at Cisco. I hold key patents on the subject. I wrote a book on the subject [14].

This experience gave a very solid background in how SLAs are handled in IP networks.

The years of designing and deploying real technology in real networks taught me that the simplest ideas prevail, and that unneeded sophistication and states leads to a lot of cost and complexity.

Some personal contributions on the QoS subject:

- Engineering a multiservice IP backbone to support tight SLAs. Computer Networks 40(1), 2002 [15]
- Deploying Diffserv at the Network Edge for Tight SLAs, Part 2. IEEE Internet Computing 8(2), 2004 [16]
- Deploying Diffserv at the Network Edge for Tight SLAs, Part 1. IEEE Internet Computing 8(1), 2004 [17]
- Deploying Diffserv in Backbone Networks for Tight SLA Control. IEEE Internet Computing 9(1), 2005 [18]
- Toward Lossless Video Transport. IEEE Internet Computing 15(6), 2011 [20]
- Multi-layer capacity planning for IP-optical networks. IEEE Communications Magazine 52(1), 2014 [19]

3.4 RSVP-TE Deployment

In the early 2000, Thomas Telkamp was managing the worldwide Global Crossing (GBLX) backbone from Amsterdam, the Netherlands. This was one of the first RSVP-TE deployments and likely the biggest at that time. I had the chance to work directly with him and his team. I learned the following concepts through the experience:

- Tactical TE is far superior than an always-on full-mesh of circuits
- ECMP is key for IP
- For the vast majority of networks, Fast Convergence (FC), QoS and Capacity Planning are the key pillars delivering SLAs, see chapter 1 of [6].

3.4.1 Rare Tactical TE is the target

"One should not need to wear raincoat and boots every day if it rains only a few days a year"

In the state-of-the-art section, we used the previous analogy to underline that tactical traffic engineering is what is required in most of cases due to capacity planning.

The outcome of this is that

- most of the traffic is correctly capacity planned and hence should be let on the IGP ECMP-aware shortest-path to its egress node
- the minority of traffic that needs to be explicitly routed should be placed on a stateless SR Policy. Most likely this requirement only exists during rare network failures not forecast by the capacity planning model

3.4.2 Data Sets

Later, under non-disclosure agreement, I worked on the largest RSVP-TE deployments and got access to key datasets. These datasets were key to shape my thought-process for the network programming project. We will come back to this later.

3.5 Fast Convergence project

From 2002 to 2009, I lead the Fast Convergence (FC) project which improved the IGP routing convergence by 2 orders of magnitude (10's of second to ~200msec) and introduced the first IP-optimized Fast Reroute techniques (Loop-Free Alternate aka LFA [21] and Remote LFA [22]) (50msec protection). I co-hold the key patents on the subject.

This experience gave a solid background in routing and especially the communication between the control-plane and the data plane and the various bottlenecks involved in updating the forwarding table. This knowledge would be key to trigger the Network Programming solution, as described in the next section.

Some personal contributions on the FC subject:

- Achieving sub-second IGP convergence in large IP networks. Computer Communication Review 35(3), 2005 [23]

- Achieving sub-50 milliseconds recovery upon BGP peering link failures. CoNEXT 2005 [25]
- Achieving sub-50 milliseconds recovery upon BGP peering link failures. IEEE/ACM Transactions on Networking 15(5), 2007 [24]

3.6 The SDN and OpenFlow Influences

In 2013, two fundamental papers were published at SIGCOMM: SWAN (Software-driven WAN) from Microsoft [26] and B4 from Google [27].

While I was agreeing with the overall idea (centralized computation and signaling is far better than the slow nondeterministic and stateful distributed circuits from RSVP-TE), I immediately saw that it would not scale for two reasons:

- The OpenFlow granularity was too thin
- The controller should focus on traffic-engineering the subset of flows that need an explicit route and leave most of the flows on their natural IGP path

3.6.1 OpenFlow Granularity

The OpenFlow granularity was way too thin.

For each flow, the controller needs to program per-flow states at every hop along the flow. Each of these FIB entries would be faced with all the bottlenecks that I had analyzed during the FC project. The system would be very complex (consistency checking of the FIB entries, error condition) and would scale poorly (number of entries to update, time to update them).

I had spent several years improving the routing convergence speed and hence I knew that the problem was not so much in the control plane and the algorithmic computation of a path, but much more in the transferring of the FIB updates from the routing processor down to the line cards and then the writing of the updates from the line card CPU to the hardware forwarding logic.

To give an order of magnitude, while the routing processor would compute and update an entry in µsec, the rest of the process (distribute to line cards, update hardware forwarding) was in msec when we started the fast convergence project. It took many years of work to get that component down to 10’s of µsec.

Hence, by intuition I would have bet that the OpenFlow-driven approach would run in severe convergence problem: it would take way too much time for the centralized control-plane to send updates to the switches and have them install in HW.

One can reverse engineer some of the numbers published in the original OpenFlow papers and realize how slow these systems were.

This was later confirmed by lead architects who worked on these deployments [28].

3.6.2 Lack of IGP ECMP-aware shortest-path

As we have seen earlier, most of the traffic just follows the IGP shortest-path. Hence, most of the controller work is spent in doing what the IGP would do which is of little benefit as the scalable IGP implementation is not a problem in 2020 (consider as well that the controller would need to perform the TILFA and µLoop avoidance for all the routes of all the routers of the network).

3.7 The intuition

Along a long drive to Rome, the following intuition came to me: when I go to Rome, from Brussels, I listen to the radio for traffic events. If I hear that the Gottardo tunnel is blocked then I mentally switch on the path to Geneva, and then from there to the path to Rome.

This intuition was: in our daily life, we do not plan our journeys turn by turn (i.e. stateful circuits); instead we plan them as a very small number of shortest-path hops. We can apply this to networking as well.

Indeed, leveraging what we introduced earlier, the following solution became clear:

- Most of the traffic should be left on the IGP ECMP-aware shortest-path.
  - This leads to the notion of a “topological shortest-path waypoint” formalized later in this document as the END segment family. This concept provides the fundamental granularity of the SR solution as opposed to the per-flow per-hop OpenFlow granularity
- A small subset of the traffic needs to be tactically traffic engineered.
  - To keep the solution scalable, I would use stateless source routing.
  - To leverage the ECMP nature of IP networks, I would use a source route of END segments.
  - To have a theoretically complete TE solution, I would have a topological shortest-path waypoint up to a node N followed by an engineered cross-connect on a specific outgoing interface of node N despite the relative routing metric of that interface. This is formalized later in the document at the END.X segments.
- The source routes would need very few segments
  - This point was clear to me because of the experience I had with real RSVP-TE deployments and the study of the data sets.
  - I demonstrated this intuition factually in [29] without being able to reveal the source of the data-sets as they were confidential.
  - In April 2019, these results were confirmed by the Google SR deployment [5].
- The solution would need to support various deployment models: distributed, centralized and hybrid
  - Distributed: this was the de-facto model in the SP industry
  - Centralized: this was the de-facto model in the WEB[1]/SDN-OpenFlow industry
  - Hybrid: I knew the shortcomings of the fully-distributed model of RSVP-TE and hence I knew that a distributed-only model would run into the same issues. Hence there was a clear need to leverage the central computation with the distributed control-plane. This research lead to the definition of the SR Policy model, SR native TE algorithms, On-Demand Next-Hop and Steering automation. We detail these in the TE section.
- The availability of a stateless and scalable explicit routing finally opened the door for a topology-independent optimum FRR solution. I had been leading the IP-FRR research for 10 years with the invention, productization and deployment of LFA [21] and Remote-LFA [22] but I knew that two

[1] The WEB market refers to networks such as Google, Amazon, Facebook, Microsoft

problems were yet to be solved: topology-independency and per-destination optimality. This research lead to the invention of the Topology-Independent Loop Free Alternate (TILFA) and the related Micro-Loop Avoidance solution. These are detailed in their respective sections.

3.8 The abstraction and the generalization

Quickly, I realized that these topological segments and these topological source routes were only a specific instantiation of a more general concept:

- A segment is generalized as a network instruction
  - The topological nature of the initial instruction idea is just an instance of the concept but not its definition. The real definition is a generic instruction that can be bound to any behavior: either any data-plane pseudocode or any container or virtual-machine.
- A list of segments is generalized as a network program
- Metadata exchanged across segments is needed
  - A program needs shared variable between its instructions

The network programming model became clear.

Historically, I first researched, productized and deployed the MPLS data plane instantiation of the solution as it was much easier to commercialize: MPLS data plane was deployed all over the SP/WEB industry and only software update would be required.

From a conceptual viewpoint, the entire scope of the model is best understood with the IPv6 data plane and hence in this document, we describe the network programming model with its IPv6 data plane.

3.9 Proposed Solution – Network Programming

The ingress edge of the Service Provider (SP) transport domain (“the source”) classifies the customer packet and encapsulates it in an outer IPv6 header with a Segment Routing Extension header (SRH).

The source controls the entire experience of the packet within the domain by encoding a stateless network program as the combination of the outer destination address (DA), the segment list and metadata in the SRH.

A segment (SID) is defined as a Locator:Function 128-bit value. The first part (locator/mask) is routed to a node in the network. Once at that node, the SID is associated with an instruction or function.

The locator of the destination address (DA) steers the packet to the next segment processing node which executes the bound function, updates the DA with the next segment and the cycle repeats until the end of the network program.

The functions may range from topological underlay or overlay instructions delivered by multi-terabit hardware engines to service-centric behaviors instantiated on CPU as entire containers or virtual machines.

These different functions can exchange data between each other thanks to the metadata TLV in the SRH.

The network programming model allows to express a wide variety of complex behaviors in an end-to-end and scalable manner through the sub-domains of the SP.

A SID may itself encode a micro-program as a list of micro-SIDs (uSID). Micro-programs guarantee scalability and efficiency of the solution even in large SP domains made of hundreds of sub-domains and hundred thousand of routers.

[30] defines the SRv6 Network Programming concept and standardizes the main segment routing functions to enable the creation of overlays with underlay optimization and service programming.

In this document, we restrict our scope to the following applications and their natural integration:

- Traffic Engineering
- 50msec Prefix-Independent Fast Reroute
- Micro-Loop Avoidance
- Optimum Load Balancing
- Overlay Virtualization
- Service Programming
- Ultra-Scale and Efficiency with uSID

The following applications are outside the scope of this document:

- OAM and Performance Monitoring
- Stateless alternative to GTP
- Container Networking
- IoT
- Security beyond the SP transport model

While the formal name of the solution is “Network Programming with Segment Routing v6”, we interchangeably use the term “Network Programming”, “Segment Routing” and most often “SRv6” or “SR” for convenience.

The solution is stateless as the state is in the packet header (i.e. the network program).

In a circuit-style solution, an intermediate router holds one state per transit flow that requires a specific processing. The state is needed to recognize the flow and then associate it with an action.

An SR intermediate node does not store any state for such flows. The classification happens via the routing to the locator part of the SID and the association of an action happens via the function part of the SID.

The stateless nature of the solution finally enables the delivery of end-to-end policies through sub-domains. Prior solutions required stateful re-classification at the edge of the sub-domains which was unscalable and very complex to operate.

Further information can be found in the selected papers [30], [38] and [31] also available in the respective annexes 1, 2 and 3.

3.10 Personal Contribution

I initiated the SR project in the industry, lead all the Cisco Systems productization and deployments, co-authored all the key IETF specifications and co-hold all the key Cisco patents on the subject. I gave the first public presentations on all aspects of the solution. I led several research projects with academia and co-authored the related scientific papers. I co-wrote the reference books on the subject.

The following sub-sections list my contributions on Segment Routing. Three of them have been selected for this thesis and are available in the annexes 1, 2 and 3.

3.10.1 SR-related co-authored Books

1. C. Filsfils, K. Michielsen, K. Talaulikar, "Segment Routing Part I". 2017.
2. C. Filsfils, K. Michielsen, F. Clad, D. Voyer, "Segment Routing Part II: Traffic Engineering". 2019.

3.10.2 SR-related co-authored Journals scientific papers

1. R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks". Computer Communication Review 45(5): 15-28 (2015).
2. P. L. Ventre, M. M. Tajiki, S. Salsano, C. Filsfils, "SDN Architecture and Southbound APIs for IPv6 Segment Routing Enabled Wide Area Networks". IEEE Trans. Network and Service Management 15(4): 1378-1392 (2018).
3. P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, "Segment Routing: a Comprehensive Survey of Research Activities, Standardization Efforts and Implementation Results," submitted paper under a second review round in IEEE Surveys and Tutorials, arXiv preprint arXiv:1904.03471v2, 2020.

3.10.3 SR-related co-authored Conferences scientific papers

1. D. Saucez, J. Kim, L. Iannone, O. Bonaventure, C. Filsfils, "A Local Approach to Fast Failure Recovery of LISP Ingress Tunnel Routers". Networking (1) 2012: 397-408.
2. R. Hartert, S. Vissicchio, P. Schaus, O. Bonaventure, C. Filsfils, T. Telkamp, P. François, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks". SIGCOMM 2015: 15-28.
3. C. Filsflis, N. K. Nainar, C. Pignataro, J. C. Cardona, P. François, "The Segment Routing Architecture". GLOBECOM 2015: 1-6.
4. A. Abdelsalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, L. Veltri, "Implementation of virtual network function chaining through segment routing in a Linux-based NFV infrastructure". NetSoft 2017: 1-5.
5. D. Lebrun, M. Jadin, F. Clad, C. Filsfils, O. Bonaventure, "Software Resolved Networks: Rethinking Enterprise Networks with IPv6 Segment Routing". SOSR 2018: 6:1-6:14.

6. A. Abdelsalam, P. L. Ventre, A. Mayer, S. Salsano, P. Camarillo, F. Clad, C. Filsfils, “Performance of IPv6 Segment Routing in Linux Kernel”. CNSM 2018: 414-419.
7. A. Abdelsalam, S. Salsano, F. Clad, P. Camarillo, C. Filsfils, “SERA: Segment Routing Aware Firewall for Service Function Chaining scenarios”. IFIP networking 2018: 46-54.
8. A. Mayer, S. Salsano, P. L. Ventre, A. Abdelsalam, L. Chiaraviglio, C. Filsfils, “An Efficient Linux Kernel Implementation of Service Function Chaining for legacy VNFs based on IPv6 Segment Routing”. Netsoft 2019: 333-341.

3.10.4 Key SR-related co-authored IETF documents

1. Segment Routing Architecture. RFC 8402 (2018)
2. SRv6 Network Programming (https://datatracker.ietf.org/doc/draft-filsfils-spring-srv6-network-programming/)
3. Segment Routing Policy Architecture (https://datatracker.ietf.org/doc/draft-ietf-spring-segment-routing-policy/)
4. IPv6 Segment Routing Header (SRH) (https://datatracker.ietf.org/doc/draft-ietf-6man-segment-routing-header/)
5. IGP Flexible Algorithm (https://datatracker.ietf.org/doc/draft-ietf-lsr-flex-algo/)
6. IS-IS Extensions to Support Routing over IPv6 Dataplane (https://datatracker.ietf.org/doc/draft-bashandy-isis-srv6-extensions/)
7. Topology Independent Fast Reroute using Segment Routing (https://datatracker.ietf.org/doc/draft-ietf-rtgwg-segment-routing-ti-lfa/)

3.10.4.1 Other SR-related co-authored IETF RFC

8. Source Packet Routing in Networking (SPRING) Problem Statement and Requirements. RFC 7855 (2016)
9. Use Cases for IPv6 Source Packet Routing in Networking (SPRING). RFC 8354 (2018)
10. Resiliency Use Cases in Source Packet Routing in Networking (SPRING) Networks. RFC 8355 (2018)
11. A Scalable and Topology-Aware MPLS Data-Plane Monitoring System. RFC 8403 (2018)

3.10.4.2 Other SR-related co-authored IETF drafts

12. SRv6 and MPLS interworking (https://datatracker.ietf.org/doc/draft-agrawal-spring-srv6-mpls-interworking/)
13. Bidirectional Forwarding Detection (BFD) for Segment Routing Policies for Traffic Engineering (https://datatracker.ietf.org/doc/draft-ali-spring-bfd-sr-policy/)
14. Segment Routing Header encapsulation for In-situ OAM Data (https://datatracker.ietf.org/doc/draft-ali-spring-ioam-srv6/)
15. OAM for Service Programming with Segment Routing(draft-ali-spring-sr-service-programming-oam-00)
16. Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6) (https://datatracker.ietf.org/doc/draft-ali-spring-srv6-oam/)
17. Packet-Optical Integration in Segment Routing (https://datatracker.ietf.org/doc/draft-anand-spring-poi-sr/)

18. IS-IS Extensions to Support Routing over IPv6 Dataplane (https://datatracker.ietf.org/doc/draft-bashandy-isis-srv6-extensions/)
19. Loop avoidance using Segment Routing (https://datatracker.ietf.org/doc/draft-bashandy-rtgwg-segment-routing-uloop/)
20. Segment Routing IPv6 for mobile user-plane PoCs (https://datatracker.ietf.org/doc/draft-camarillo-dmm-srv6-mobile-pocs/)
21. SRv6 Mobility Use-Cases (https://datatracker.ietf.org/doc/draft-camarilloelmalky-springdmm-srv6-mob-usecases/)
22. BGP-LS Advertisement of Segment Routing Service Segments (https://datatracker.ietf.org/doc/draft-dawra-idr-bgp-ls-sr-service-segments/)
23. BGP Link State extensions for IPv6 Segment Routing(SRv6) (https://datatracker.ietf.org/doc/draft-dawra-idr-bgpls-srv6-ext/)
24. BGP Signaling for SRv6 based Services. (https://datatracker.ietf.org/doc/draft-dawra-idr-srv6-vpn/)
25. Comparative Analysis of MTU overhead in the context of SPRING (https://datatracker.ietf.org/doc/draft-dukes-spring-mtu-overhead-analysis/)
26. SR For SDWAN: VPN with Underlay SLA (https://datatracker.ietf.org/doc/draft-dukes-spring-sr-for-sdwan/)
27. Interconnecting Millions Of Endpoints With Segment Routing (https://datatracker.ietf.org/doc/draft-filsfils-spring-large-scale-interconnect/)
28. SR Policy Implementation and Deployment Considerations (https://datatracker.ietf.org/doc/draft-filsfils-spring-sr-policy-considerations/)
29. SRv6 interoperability report (https://datatracker.ietf.org/doc/draft-filsfils-spring-srv6-interop/)
30. Illustrations for SRv6 Network Programming (https://datatracker.ietf.org/doc/draft-filsfils-spring-srv6-net-pgm-illustration/)
31. SRv6 Network Programming (https://datatracker.ietf.org/doc/draft-filsfils-spring-srv6-network-programming/)
32. Segment Routing with MPLS Data Plane encapsulation for In-situ OAM Data (https://datatracker.ietf.org/doc/draft-gandhi-spring-ioam-sr-mpls/)
33. In-band Performance Measurement for Segment Routing Networks with MPLS Data Plane (https://datatracker.ietf.org/doc/draft-gandhi-spring-rfc6374-srpm-mpls/)
34. In-band Performance Measurement Using UDP Path for Segment Routing Networks (https://datatracker.ietf.org/doc/draft-gandhi-spring-rfc6374-srpm-udp/)
35. In-band Performance Measurement Using TWAMP for Segment Routing Networks (https://datatracker.ietf.org/doc/draft-gandhi-spring-twamp-srpm/)
36. IPv6 Segment Routing Header (SRH) (https://datatracker.ietf.org/doc/draft-ietf-6man-segment-routing-header/)
37. Segment Routing IPv6 for Mobile User Plane (https://datatracker.ietf.org/doc/draft-ietf-dmm-srv6-mobile-uplane/)
38. Revised Validation Procedure for BGP Flow Specifications (https://datatracker.ietf.org/doc/draft-ietf-idr-bgp-flowspec-oid/)
39. BGP Link-State extensions for Segment Routing (https://datatracker.ietf.org/doc/draft-ietf-idr-bgp-ls-segment-routing-ext/)

40. Segment Routing Prefix SID extensions for BGP (https://datatracker.ietf.org/doc/draft-ietf-idr-bgp-prefix-sid/)
41. BGP-LS extensions for Segment Routing BGP Egress Peer Engineering (https://datatracker.ietf.org/doc/draft-ietf-idr-bgpls-segment-routing-epe/)
42. Advertising Segment Routing Policies in BGP (https://datatracker.ietf.org/doc/draft-ietf-idr-segment-routing-te-policy/)
43. BGP-LS Advertisement of IGP Traffic Engineering Performance Metric Extensions (https://datatracker.ietf.org/doc/draft-ietf-idr-te-pm-bgp/)
44. Advertising L2 Bundle Member Link Attributes in IS-IS (https://datatracker.ietf.org/doc/draft-ietf-isis-l2bundles/)
45. Signaling Entropy Label Capability and Entropy Readable Label Depth Using IS-IS (https://datatracker.ietf.org/doc/draft-ietf-isis-mpls-elc/)
46. IS-IS Extensions for Segment Routing (https://datatracker.ietf.org/doc/draft-ietf-isis-segment-routing-extensions/)
47. IGP Flexible Algorithm (https://datatracker.ietf.org/doc/draft-ietf-lsr-flex-algo/)
48. Signaling Entropy Label Capability and Entropy Readable Label-stack Depth Using OSPF (https://datatracker.ietf.org/doc/draft-ietf-ospf-mpls-elc/)
49. OSPF Extensions for Segment Routing (draft-ietf-ospf-segment-routing-extensions-27)
50. Path Computation Element Communication Protocol (PCEP) Extensions for remote-initiated GMPLS LSP Setup (https://datatracker.ietf.org/doc/draft-ietf-pce-remote-initiated-gmpls-lsp/)
51. PCEP Extensions for Segment Routing (https://datatracker.ietf.org/doc/draft-ietf-pce-segment-routing/)
52. BGP Prefix Independent Convergence (https://datatracker.ietf.org/doc/draft-ietf-rtgwg-bgp-pic/)
53. Topology Independent Fast Reroute using Segment Routing (https://datatracker.ietf.org/doc/draft-ietf-rtgwg-segment-routing-ti-lfa/)
54. Segment Routing Centralized BGP Egress Peer Engineering (https://datatracker.ietf.org/doc/draft-ietf-spring-segment-routing-central-epe/)
55. Segment Routing interworking with LDP (https://datatracker.ietf.org/doc/draft-ietf-spring-segment-routing-ldp-interop/)
56. Segment Routing with MPLS data plane (https://datatracker.ietf.org/doc/draft-ietf-spring-segment-routing-mpls/)
57. BGP-Prefix Segment in large-scale data centers (https://datatracker.ietf.org/doc/draft-ietf-spring-segment-routing-msdc/)
58. Segment Routing Policy Architecture (https://datatracker.ietf.org/doc/draft-ietf-spring-segment-routing-policy/)
59. Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) extension for recording TE Metric of a Label Switched Path (https://datatracker.ietf.org/doc/draft-ietf-teas-te-metric-recording/)
60. BGP Link-State Extensions for BGP-only Fabric (https://datatracker.ietf.org/doc/draft-ketant-idr-bgp-ls-bgp-only-fabric/)
61. LISP Control Plane for SRv6 Endpoint Mobility (https://datatracker.ietf.org/doc/draft-rodrigueznatal-lisp-srv6/)

62. Carrying Binding Label/Segment-ID in PCE-based Networks (https://datatracker.ietf.org/doc/draft-sivabalan-pce-binding-label-sid/)
63. Insertion of IPv6 Segment Routing Headers in a Controlled Domain (https://datatracker.ietf.org/doc/draft-voyer-6man-extension-header-insertion/)
64. SR Replication Policy for P2MP Service Delivery (https://datatracker.ietf.org/doc/draft-voyer-spring-sr-p2mp-policy/)
65. Service Programming with Segment Routing (https://datatracker.ietf.org/doc/draft-xuclad-spring-sr-service-programming/)

3.10.5 SR-related co-authored public Patents

Public patents (~40) related to this project can be found here: https://patents.google.com/?q=segment&q=routing&inventor=Clarence+Filsfils&page=4

More are in filing process and hence not public yet.

Some of the key SR patents are referenced in the below table.

Reference	**Title and url to USA Patent Application**	**USPTO #**
[Patent-Base]	Base SR Technique	20140169370
[Patent-Net-Pgm]	Enhanced SR processing of a packet	20180375766
[Patent-SRH]	Segment routing extension headers	9762488B2
[Patent-TE-ODN]	On-Demand Next-Hop Resolution	20170230274
[PATENT-TACTICAL-TE]	Tactical traffic engineering based on segment routing policies	US10212088
[Patent-VPN]	Ethernet Virtual Private Network (EVPN) using an Internet Protocol Version 6 Segment Routing (SRv6) Underlay Network and SRv6-enhanced Border Gateway Protocol (BGP) Signaling	20180375763
[Patent-Serv-Pgm]	Segment Routing Gateway Storing Segment Routing Encapsulating Header Used in Encapsulating and Forwarding of Returned Native Packet	20180375684

3.11 List of publications selected for this thesis

C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir. 2018. “Segment Routing Architecture.” RFC Editor. This article is included in Annex 1 and is referenced as [31]. Annex 1.

C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer. 2019. “IPv6 Segment Routing Header (SRH).” Internet Engineering Task Force. RFC editor. This article is included in Annex 2 and is referenced as [38]. Annex 2.

C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li. 2019. “SRv6 Network Programming.” http://www.ietf.org/internet-drafts/draft-ietf-spring-srv6-network-programming. Last-call process. This article is included in Annex 3 and is referenced as [30]. Annex 3.

4 SEGMENT ROUTING

This section describes the Segment Routing architecture [31] (annex 1), the SR Extension Header [38] (Annex 2) and the network programming model [30] (Annex3).

In the first section, we introduce the segment routing architecture.

Section 2, describes the network programming model and the concepts of instruction, function and behavior.

Section 3 describes the segment routing extension header.

In the fourth and fifth sections, we respectively describe how the SR domain is secured and how optimal load-balancing is delivered within the SR domain.

In the sixth and seventh sections, we respectively describe Topology-Independent Loop Free Alternate and Micro-Loop Avoidance. These are two fundamental SR innovations related to service availability despite routing transitions.

Section 8 and 9 respectively describe the integration of Overlay Virtualization (VPN) and service chaining in the network program.

Section 10 describes the scaling property of the solution.

4.1 Segment Routing Architecture

Segment Routing (SR) leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called "segments". A segment is often referred to by its Segment Identifier (SID) and can represent any instruction, topological or service based. A segment can have a semantic local to an SR node or global within an SR domain.

SR domain is the set of nodes participating in the source-based routing model and managed by the same administrative entity. These nodes may be connected to the same physical infrastructure (e.g., a Service Provider's network). They may as well be remotely connected to each other (e.g., an enterprise VPN or an overlay). SR supports per-flow explicit routing while maintaining per-flow state only at the ingress nodes to the SR domain.

A segment may be associated with a topological instruction. A topological local segment may instruct a node to forward the packet via a specific outgoing interface. A topological global segment may instruct an SR domain to forward the packet via a specific path to a destination. Different segments may exist for the same destination, each with different path objectives (e.g., which metric is minimized, what constraints are specified).

A segment may be associated with a service instruction (e.g., the packet should be processed by a container or Virtual Machine (VM) associated with the segment). A segment may be associated with a QoS treatment (e.g., shape the packets received with this segment at x Mbps).

The SR architecture supports any type of instruction associated with a segment.

The SR architecture supports any type of control plane: distributed, centralized, or hybrid. In a distributed scenario, the segments are allocated and signaled by IS-IS or OSPF or BGP. A node individually decides to steer packet on an SR Policy. A node individually computes the SR Policy.

In a centralized scenario, the segments are allocated and instantiated by an SR controller. The SR controller decides which nodes need to steer which packets on which source-routed policies. The SR controller computes the source-routed policies. The SR architecture does not restrict how the controller programs the network. Likely options are Network Configuration Protocol (NETCONF), Path Computation Element Communication Protocol (PCEP), and BGP. The SR architecture does not restrict the number of SR controllers. Specifically, multiple SR controllers may program the same SR domain.

A hybrid scenario complements a base distributed control plane with a centralized controller. For example, when the destination is outside the IGP domain, the SR controller may compute an SR Policy on behalf of an IGP node. The SR architecture does not restrict how the nodes that are part of the distributed control plane interact with the SR controller. Likely options are PCEP and BGP.

The SR architecture can be instantiated on various data planes. In this thesis, we focus on the SR instantiation over the IPv6, knows as SRv6. SR is applied to the IPv6 architecture with a new type of routing header called the SR Header (SRH). An instruction is associated with a segment and encoded as an IPv6 address. An SRv6 segment is also called an SRv6 SID. An SR Policy is instantiated as an ordered list of SRv6 SIDs in the routing header. The active segment is indicated by the Destination Address (DA) of the packet. The next active segment is indicated by the SegmentsLeft (SL) pointer in the SRH. When an SRv6 SID is completed, the SL is decremented and the next segment is copied to the DA. When a packet is steered on an SR Policy, the related SRH is added to the packet.

4.2 A program as a Segment List

A network program is represented as a SID list <S1, S2…, Sk> where S1 is the first SID to visit, S2 is the second SID to visit and Sk is the last SID to visit along the SR path.

The network program of a packet is encoded in the outer DA and the SRH.

4.2.1 SRv6 Segment

An SRv6 Segment is a 128-bit value. "SID" (abbreviation for Segment Identifier) is often used as a shorter reference for "SRv6 Segment".

An SRv6-capable node N maintains a "My SID Table". This table contains all the SRv6 segments explicitly instantiated at node N and binds each of them with a specific function. N is the parent node for these SIDs.

We represent an SRv6 SID as LOC:FUNCT where LOC is the L most significant bits and FUNCT is the 128-L least significant bits. L is called the locator length and is flexible. Each operator is free to use the locator length it chooses. Most often the LOC part of the SID is routable and leads to the node which instantiates that SID.

The FUNCT part of the SID is an opaque identification of a local function bound to the SID.

A function may require additional arguments that would be placed immediately after the FUNCT. In such case, the SRv6 SID will have the form LOC:FUNCT:ARGS::. The usage of the argument will be illustrated in the uSID section.

A node may receive a packet with an SRv6 SID in the DA without an SRH. In such case the packet should still be processed by the Segment Routing engine. This represents a network program made of one single instruction.

4.2.2 Function associated with a SID

Each entry of the "My SID Table" indicates the function associated with the local SID.

The END and END.X functions are defined in this section.

The T.ENCAPS.RED behavior is defined in the TE section.

The END.DX6 function is defined in the Overlay section.

The application proxy functions are defined in the service programming section.

For readability, we defer the explanation of the uSID and micro-program to a later section.

4.2.3 END - Endpoint

The Endpoint function ("End" for short) is the most basic function.

It provides an ECMP-aware shortest-path to the segment “endpoint”. Transit nodes along this path act as classic IPv6 forwarders.

When the packet reaches the “endpoint”, the “endpoint” checks whether there is an SRH (next header field NH of the outer Ipv6 header) and whether there are other segments left (SL field) in the SRH. If so, it copies the next segment in the destination address, decrements the SegmentLeft (SL) field of the SRH and forwards the packet as per the updated destination address. Else it drops the packet as this specific function is used as an intermediate topological waypoint and not a final destination.

When N receives a packet whose IPv6 DA is S and S is a local End SID, N executes:

```
1.   IF NH=SRH and SL > 0
2.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      FIB lookup on the updated DA
5.      forward according to the matched entry
6.   ELSE IF NH!=SRH
7.      Send an ICMP parameter problem message; drop the packet
8.   ELSE
9.      drop the packet
```

At line 4, The End function performs the FIB lookup in the forwarding table associated with the ingress interface. A typical deployment would have one END SID per node.

4.2.4 END.X - Endpoint with cross-connect to an array of layer-3 adjacencies

This function is the same as the previous one with one exception: instead of looking up the updated destination address and forwarding accordingly, the packet is “cross-connected” on an array of interfaces bound to the END.X SID.

An operator may set a high IGP metric on a link to prevent any shortest-path traffic on the link (e.g. the link is only used for worst-case partition failure or maybe this is an excellent low-latency link but whose capacity needs to be preserved for the sole latency traffic).

The END.X SID associated with such a link enables an SR policy to bring the packet up to the parent node and then force the packet through the desired link, independently of its IGP metric.

When N receives a packet destined for S and S is a local End.X SID, N executes:

```
1.    IF NH=SRH and SL > 0
2.       decrement SL
3.       update the IPv6 DA with SRH[SL]
4.       forward to layer-3 adjacency bound to the SID S
6.    ELSE IF NH!=SRH
7.       Send an ICMP parameter problem message; drop the packet
8.    ELSE
9.       drop the packet
```

At line 4, if an array of adjacencies is bound to the End.X SID, then one entry of the array is selected based on a hash of the packet’s header.

4.2.5 Behavior Identification

When a SID S is instantiated at Node N, S is bound to a specific function by specifying a behavior ID. Table 4 of [30] maintains the registry of the defined SR behaviors. This is a key resource to ensure inter-operability between vendors and hence a healthy ecosystem.

4.2.6 Control Plane

In an SDN environment, one expects the controller to explicitly provision the SIDs and/or discover them as part of a service discovery function. Applications residing on top of the controller could then discover the required SIDs and combine them to form a distributed network program.

The concept of "SRv6 network programming" refers to the capability for an application to encode any complex program as a set of individual functions distributed through the network. Some functions relate to underlay SLA, others to overlay/tenant, others to complex applications residing in VM and containers.

4.2.6.1 IGP

The End and End.X SIDs are signaled in the IGP [35]

These SIDs provide important topological functions for the IGP to build TI-LFA and µLoop solutions and for TE to build SR policies as explained later.

4.2.6.2 BGP-LS

BGP-LS is the key service discovery protocol. In a nutshell, BGP-LS uses the BGP protocol to convey topological and service information from a routing node to a collecting server. The BGP protocol is convenient as this is a key protocol maintained and available on the router. It scales well and it allows multiple collecting servers to receive the information in parallel. Any information can be encoded. For the scope of this document, the two key pieces of information are: topological data (reconstruct the network graph of the domain together with its characteristics (metrics, affinities…)), service data (which services are available at what node with what characteristics) and SID information (topological and service).

Every node is expected to advertise via BGP-LS its SRv6 capabilities (e.g. how many SIDs it can insert as part of a T.Insert behavior) and any locally instantiated SID [36].

4.2.6.3 BGP IP/VPN/EVPN

The End.DX/DT SIDs are signaled in BGP [37]. We will formally define them in the overlay section.

4.2.6.4 TE

This is detailed in the TE section.

4.3 Segment Routing Header

This section describes the Segment Routing Extension Header (SRH) and how it is used by Segment Routing capable nodes.

Further information can be found in the selected papers [38] and [39] also available in appendix 2.

4.3.1 Header Format

The Segment Routing Header (SRH) is defined as follows:

- Next Header, Hdr Ext Len and Segments Left: defined in [40] Section 4.4
- Routing Type: TBD, to be assigned by IANA (suggested value: 4)
- Last Entry: contains the index (zero based) in the Segment List of the last element of the Segment List
- Flags: 8 bits of flags
- Tag: tag a packet as part of a class or group of packets, e.g., packets sharing the same set of properties.
- Segment List[n]: 128-bit IPv6 address representing the nth segment in the Segment List. The Segment List is encoded starting from the last segment of the SR Policy, i.e., the first element of the segment list (Segment List [0]) contains the last segment of the SR Policy, the second element contains the penultimate segment of the SR Policy and so on.
- Type Length Value (TLV): provides meta-data for segment processing (e.g. service programming, see section 7.2 of [41])

4.3.2 Hardware Performance

The SR header has been optimized for hardware performance: i.e. the metadata TLVs are placed after the SID list.

Metadata TLVs are mostly applicable for CPU-centric service functions and hence should not impact the performance of the hardware read in the segment list.

4.3.3 SR Nodes

There are different types of nodes that may be involved in segment routing networks: source SR nodes originate packets with a segment in the destination address of the IPv6 header, transit nodes that forward packets destined for a remote segment, and SR segment endpoint nodes that process a local segment in the destination address of an IPv6 header.

4.3.4 Packet Processing

We describe SRv6 packet processing at the SR source, Transit and SR segment endpoint nodes.

4.3.4.1 Source SR Node – T.Encaps

A Source node steers a packet into an SR Policy. If the SR Policy results in a segment list containing a single segment, and there is no need to add information to SRH flag or TLV, the DA is set to the single segment list entry and the SRH MAY be omitted.

When needed, the SRH is created as follows:

- Next Header and Hdr Ext Len fields are set as specified in [40]
- Routing Type field is set as TBD (to be allocated by IANA, suggested value 4)
- The DA of the packet is set with the value of the first segment
- The first element of the SRH Segment List is the ultimate segment
- The second element is the penultimate segment and so on
- The Segments Left field is set to n-1 where n is the number of elements in the SR Policy
- The Last Entry field is set to n-1 where n is the number of elements in the SR Policy

The packet is forwarded toward the packet's Destination Address (the first segment).

This is formally defined in [30] as the T.Encaps/T.Insert behaviors.

4.3.4.1.1 Reduced SRH

When a source does not require the entire SID list to be preserved in the SRH, a reduced SRH may be used.

A reduced SRH does not contain the first segment of the related SR Policy (the first segment is the one already in the DA of the IPv6 header), and the Last Entry field is set to n-2 where n is the number of elements in the SR Policy.

This is formally defined in [30] as the T.Encaps.Reduced/T.Insert.Reduced behaviors.

4.3.4.2 Transit Node

As specified in [40], the only node allowed to inspect the Routing Extension Header (and therefore the SRH), is the node corresponding to the DA of the packet. Any other transit node MUST NOT inspect the underneath routing header and MUST forward the packet toward the DA according to its IPv6 routing table.

When a SID is in the destination address of an IPv6 header of a packet, it is routed through an IPv6 network as an IPv6 address. SIDs, or the prefix(es) covering SIDs, and their reachability may be distributed by an IGP or BGP.

4.3.4.3 SR Segment Endpoint Node

An SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packet destination address. This lookup can return any of the following:

- A FIB entry that represents a locally instantiated SRv6 SID
- A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID
- A FIB entry that represents a non-local route
- No Match

The resulting action in the last two cases is not changed by the SRH specification: i.e. respectively forward along the matched route and drop.

The two other actions are detailed in the next sections.

4.3.4.3.1 FIB Entry Is Locally Instantiated SRv6 SID

If the FIB entry is a locally instantiated SRv6 SID, then the bound behavior is executed: e.g. END or END.X as defined in the previous section.

Section 4.3.1 of [38] provides a detailed framework for the specification of the “behavior” attached to a local SID, taking into account various error/consistency verifications.

4.3.4.3.2 FIB Entry is a Local Interface not instantiated as an SRv6 SID

If the FIB entry represents a local interface, not locally instantiated as an SRv6 SID, the SRH is processed as follows:

- If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing Header
- If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet’s Source Address, pointing to the unrecognized Routing Type

4.3.5 Abstract Representation of an SRH

This section describes the abstract SRH representation that we devised to illustrate the journey of packets through a network program while omitting the non-essential parts of the SRH.

For a node k, its IPv6 address is represented as AK, its SRv6 SID is represented as SK.

IPv6 headers are represented as the tuple of (source, destination). For example, a packet with source address A1 and destination address A2 is represented as (A1,A2). The payload of the packet is omitted.

An SR Policy is a list of segments. A list of segments is represented as <S1,S2,S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit.

(SA, DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- Source Address is SA, Destination Addresses is DA, and next-header is SRH
- SRH with SID list <S1, S2, S3> with SegmentsLeft = SL

Note the difference between the <> and () symbols. <S1, S2, S3> represents a SID list where the leftmost segment is the first segment. Whereas, (S3, S2, S1; SL) represents the same SID list but encoded in the SRH Segment List format where the leftmost segment is the last segment. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of detailed behavior, the (S3, S2, S1; SL) notation is more convenient.

At its SR Policy headend, the Segment List <S1,S2,S3> results in SRH (S3,S2,S1; SL=2) represented fully as:

- Segments Left=2
- Last Entry=2
- Flags=0
- Tag=0
- Segment List[0]=S3
- Segment List[1]=S2
- Segment List[2]=S1

For brevity, we have decided to not include any example in this document. Instead, we provide references to the IETF specifications that we co-authored.

4.4 Securing the SR Domain

Nodes outside the SR Domain are not trusted: they cannot directly use the SIDs of the domain. This is enforced by two levels of access control lists.

First level: domain-wide: any packet entering the SR Domain and destined for a SID within the SR Domain is dropped.

This may be realized with the following logic, other methods with equivalent outcomes are considered compliant

- allocate all the SIDs in the domain from an address block S/s
- configure each external interface of each edge node of the domain with an inbound infrastructure access list (IACL) which drops any incoming packet with a destination address in S/s

Failure to implement this method of ingress filtering exposes the SR Domain to source routing attacks as described in [8].

Second level: node-wide: a node should drop packets to SIDs from source addresses outside the SR Domain.

This may be realized with the following logic:

- assign all interface addresses in the domain from prefix A/a
- at node k, all SIDs local to k are assigned from prefix Sk/sk
- configure each internal interface of each SR node k in the SR Domain with an inbound IACL which drops any incoming packet with a destination address in Sk/sk if the source address is not in A/a

Further information can be found in the selected paper [38] also available in the appendix.

The security aspect of the SR solution applied outside of the SR domain is outside the scope of this document.

4.5 Optimum Load-Balancing

IP networks are meshed with many ECMP path from A to Z. Optimum load-balancing is essential to efficient IP operation.

The SRv6 solution provides optimum load-balancing (LB) as it requires that

- an ingress edge node of the SP computes a hash (flow entropy) on the inner packet header and writes the result in the outer Flow Label
- an intermediate node includes the outer Flow Label in its hash to select a forwarding path out of an ECMP set

While this LB solution has not been innovated by the SR project [42], the SR project made it a MUST requirement for any node claiming SR support.

Further information can be found in the selected paper [38] also available in the appendix.

The MPLS load-balancing is sub-optimum because an interior node of the domain must resort to deep-packet inspection to find the transported header entropy. History shows that this is still difficult 20 years after MPLS design. When the hardware does not get to the transported header entropy, it computes the load-balancing hash on a subset of the MPLS label stack. This entropy is much smaller than the one of the transported header (aggregation towards common inter-domain border routers or common egress edge routers or common service) and hence there is a risk that the traffic subject to load-balancing be distributed among the ECMP paths in an unbalanced manner. As this inefficiency repeats across routers and flows, this may create network congestion that is not expected by the capacity planning model of the operator (as the capacity planning model assumes optimum load-balancing traffic dispersion). Aside the economic impact to the service-level-agreement negotiated with customers, this creates operational costs as the issue is very hard to troubleshoot or correct.

4.6 Topology-Independent Loop Free Alternate (TILFA)

An intermediate node N (called the point of local repair, PLR) on the path to a destination D may precompute a per-destination backup path B for D in advance of a future failure of a local resource R. R can be the primary link to D or the primary node to D or a local SRLG on the path to D.

There are two key novelties that we brought with Segment Routing with respect to the state of the art: the optimality of the backup path and the topology independence.

We called the solution Topology-Independent Loop Free Alternate (TILFA) and we specified it at the IETF in the selected paper [61], also available in the appendix. We filed several patents on the solution.

On a per destination D basis, the PLR does the following:

- Call the current topology T
- Determine the resource R that could fail
- Build T(R) the topology minus the resource R
- Compute the shortest-path to D in T(R)
- Encode it as an as-short-as-possible non-looping SID list
    - Recursively ensure that the shortest-path in topology T from the previous SID to the next SID does not include the PLR. Start the recursion from the PLR.
    - Stop at the earliest possible SID such that the shortest-path in topology T from that SID to D avoids PLR.
- Install it in the forwarding table as a backup path to D

This solution provides an optimal backup path as this is the post-convergence path to D from the PLR.

In the prior-art technology that we co-invented (LFA, RLFA), we were unable to guarantee the post-convergence path as we did not have a stateless and scalable explicit routing solution. As a result, it was likely that three routing paths would be seen for each failure: the optimum path before the failure, a backup path that restores the connectivity but that is not along the shortest path and finally the shortest path once the IGP has converged. This meant more jitter for the application and potentially loss due to capacity planning issue: the traffic not using the expected shortest path traverses links that have not been sized for that traffic. See [61] for a description of this issue of the prior art.

The solution is topology-independent because we can compute it in any topology.

The prior-art solution was not guaranteed to find a backup path. When we designed that prior-art solution, we wrote [62] to report typical coverage in SP topologies and topology design techniques to maximize their coverage. These constraints are lifted thanks to TI-LFA.

SR-based TILFA is now largely deployed.

These deployments confirmed the simulations (section 8 of [61]) that we had done when inventing the solution: few SIDs are required to encode the post-convergence path.

Figure 1 presents the simulations I had presented at MPLS World Congress 2016.

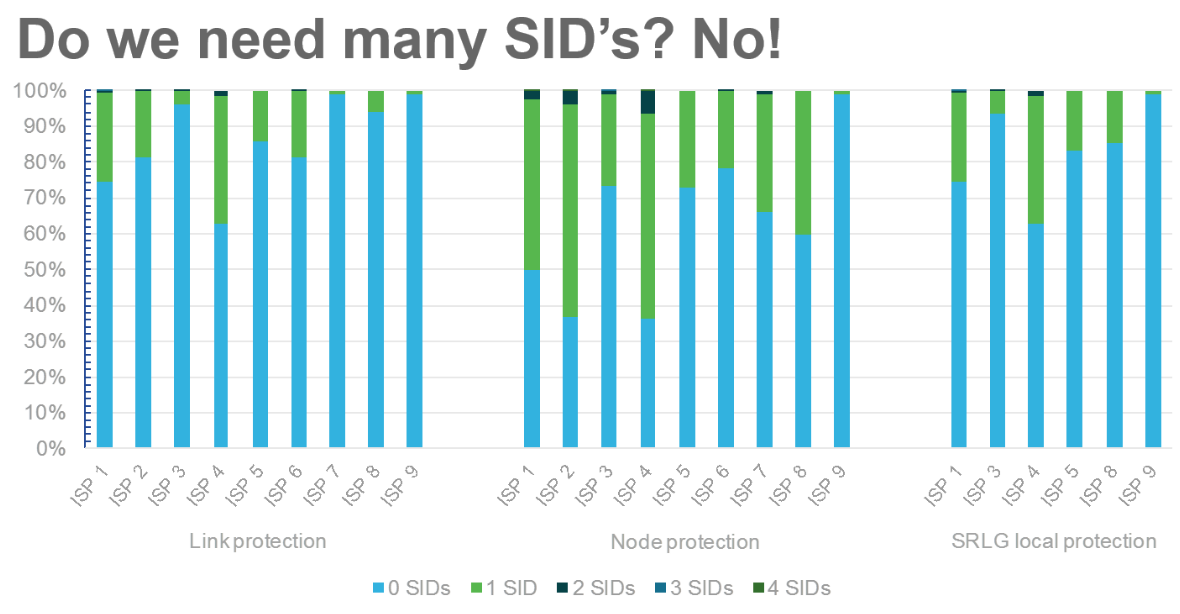


Figure 1: TILFA Scalability

4.7 Micro-Loop Avoidance

Transient forwarding loops (called micro-loop or µLoop) happen during the convergence of the IGP, because of inconsistency among forwarding states of the nodes of the network.

While we had been researching on this problem since 2003, prior to SR, the best solution we had found [63] was still fairly limited: link-down event, non-deterministic and for destinations with precomputed backup paths.

We co-authored the SR-based µLoop IETF specification [64] and several patents on the subject.

We provide hereafter a summary of its behavior and benefits.

4.7.1 Stateless Explicit Path Capability

Using segment routing, a headend can enforce an explicit path without creating any state along the desired path. As a result, a converging node can enforce traffic on the post-convergence path in a loop-free manner, using a list of segments (typically short).

4.7.2 Loop-free two-stage convergence process

Upon detecting a topology change, a node R first assesses whether the event has an impact on its forwarding paths to any destination. If it has, node R computes its post-convergence forwarding graph, then walks that graph to determine a guaranteed loop-free SID-list for each destination.

Each destination for which an empty SID-list was produced is installed normally in the forwarding table, with the outgoing interface and first hop information. These destinations are not at risk of micro-loop.

For every other destination D, associated with a non-empty SID-list, node R applies the following two-stage convergence process.

Stage 1: For a predetermined amount of time C, R installs a FIB entry for D that steers packets to D via the associated loop-free SID list.

C should be greater than or equal to the worst-case convergence time of a node, network-wide. In practice, a few seconds to be conservative.

Stage 2: After C elapses, R installs the normal post-convergence FIB entry for D, i.e. without any additional segments inserted that ensure the loop-free property.

Loop-freeness is ensured during this process, because:

- Paths made entirely of non-up-to-date routers are loop-free
  - Routers which forward as per the initial state of the network are consistent
  - Reminder: we assume TI-LFA protection in case of resource failure; hence the initial state always provides connectivity.
- A packet reaching a node in stage 1 is ensured to reach its destination in a loop-free manner
  - When a packet reaches a router in stage 1, it is steered on a SR path that enforces the post-convergence path, regardless of the state of other routers on the path.
- Paths made of a mix of routers in stage 1 and stage 2 are consistent.
  - All routers are forwarding as per their post-convergence paths, either expressed classically or as a loop-free SR path.

4.7.3 Benefits

Much wider applicability than the prior art:

- Local and remote events
- Link down and link up event
- Metric increase/decrease event

Seamless Deployment:

- No requirement for a full network upgrade: nodes that are not upgraded to the SR µloop technology will cause the µloops that they would have otherwise caused
- Incremental benefit every time a node supports SR µloop: either µloops are entirely avoided or reduced
- No protocol extension: this is a local node behavior

Capacity planning friendly:

- We leverage the post-convergence path as much as possible and hence the capacity planned path.

4.8 Overlay Virtualization

The SR solution provides overlay virtualization (VPN).

Further information can be found in the selected paper [37], [30], [65], [66] and [67] (also available in the appendix).

We provide hereafter the key components of the solution.

4.8.1 VPN SID

A simple example of a VPN SID is End.DX6: it brings the inner customer packet to the egress PE (locator of the SID). The egress PE removes the outer header and forwards the inner customer packet on a link bound with the SID. No FIB lookup in a per-customer forwarding table is required.

End.DX6 (Decapsulation and IPv6 cross-connect) is defined as follows:

```
When N receives a packet destined for S and S is a local End.DX6 SID,
N executes:

1.   IF NH=SRH and SL > 0
2.      drop the packet
3.   ELSE IF ENH = 41
4.      pop the (outer) IPv6 header and its extension headers
5.      forward to layer-3 adjacency bound to the SID S
6.   ELSE
7.      Send an ICMP parameter problem message                ;; Ref4
8.      drop the packet
```

Several other VPN SIDs are defined in [30]: End.DX4/2, End.DT6/4/46…

4.8.2 Overlay with underlay SLA

The natural integration of overlay and underlay SLA as a single network program (hence SID list) is a key novelty of our solution and a major deployment driver for the operator community: SP, WEB, and Enterprise[2].

We also provide for an automation of this integration and call it ODN/AS: “On-Demand Next-hop and Automated Steering”. This solution is of specific interest for the SP/Enterprise community as it fits their distributed control-plane dependency (e.g. BGP and IGP).

BGP routes provide reachability to services: Internet, L3VPN, PW, L2VPN.

We allow the operator to mark the BGP routes with colors. Hence, any BGP route has a next-hop and a color.

The next-hop indicates where we need to go.

The color is an extended color attribute expressed as 32-bit values. The color indicates how we need to go the next-hop. It defines a TE SLA intent.

An operator allocates TE SLA intent to each color as he wishes. A likely example is:

[2] Networks operated within corporations: e.g. Walmart.

- No color: “best-effort”
- Red: “low-delay”

Let us assume the operator marks a BGP route 9/8 via 1.1.1.5 with color red while BGP route 8/8 via 1.1.1.5 is left uncolored.

Upon receiving the route 8/8, Node 1 installs 8/8 via the END.DX4 segment associated with the route. This is the classic behavior. The traffic to 8/8 takes the IGP path to 1.1.1.5 which is the best-effort (lowest cost) path.

Upon receiving the route 9/8, Node 1 detects that the route has a color red that matches a local TE SLA template red.

> A template specifies which metric needs to be minimized and what constraints need to be taken into account (e.g. exclude some link-affinity or SRLG). Each color is mapped to a template. Each PE participating to the ODN/AS solution is pre-configured with these templates. This is operationally simple because there are few colors/templates and all the PE’s are configured with the same (few) templates. This is further simplified with network automation operational processes.

Having matched a local SLA template, the BGP process asks the TE process for the local SR Policy with (color = red; endpoint = 1.1.1.5). If this SR Policy does not yet exist, then the TE process instantiates it on-demand. Whether pre-existing or on-demand instantiated, the TE process eventually returns an SR Policy to the BGP process. The BGP process then installs 9/8 on the union of the returned SR policy and the END.DX4 segment associated with the VPN route. The traffic to 9/8 takes the red SR Policy to 1.1.1.5 which provides the low-delay path to node 5.

The installation of a BGP route onto an SR Policy is called Automated Steering (AS). This is fundamental simplification as one no longer needs to resort to complex policy-based routing constructions.

The dynamic instantiation of an SR Policy based on a color template and an endpoint is called On-Demand Next-hop (ODN). This is a fundamental simplification as one no longer needs to pre-configure any SR Policy. Instead, all the edge nodes are configured with the same few templates.

The AS and ODN functionalities have been shipping on the Cisco product line since 2015 and have been a major deployment motivation. Several patents have been filed on the topic.

4.8.3 BGP Egress Peer Engineering

The BGP Egress Peer Engineering (BGP-EPE) solution is another example of integration between the overlay and the underlay. It fits the WEB market and its centralized control plane.

We defined the BGP-EPE problem statement in [68], its solution in [69] and its BGP-LS protocol extension in [53]. We filed patent on the topic.

A centralized controller should be able to instruct an ingress Provider Edge router (PE) or a content source within the domain to use a specific egress PE and a specific external interface/neighbor to reach a destination. This SR policy is called a BGP-EPE policy.

In its most basic form, a BGP-EPE policy involves an END.X SID: shortest-path to the chosen egress PE and then cross-connect on the chosen egress peering interface.

Obviously, underlay TE segments can be inserted at the start of the policy to express an integrated underlay and overlay traffic engineering.

Several types of BGP peering segments have been defined. The cross-connect behavior can be associated with a specific interface, or a peer (and hence any interface to the peer) or a set of peers (likely from the same AS).

The EPE solution has attracted much interest. It enables content providers to monitor the quality of various peers to reach a given user and then to steer content over the selected peer (e.g. Facebook's Edge Fabric [70] and Google's Espresso [71]).

4.8.4 Enterprise policy applicability

In [67], we extended this benefit to the Enterprise policy use-case. DNS extensions enable the application to request the required behavior (underlay SLA, service program, specific exit point to the Internet) from the network. The DNS controller leverages the SDN controller to translate the required network behavior in a SID list. The DNS controller then updates the application.

In [72], we applied this benefit to the SDWAN use-case and proposed the related LISP extensions [73].

4.8.5 Control-Plane extension

There are two dominant overlay signaling protocols: BGP and LISP. We co-authored the SR extensions for both protocols.

The BGP extension is straightforward: the BGP Prefix-SID attribute [74] carries the SRv6 SID in newly defined TLV.

To preserve BGP packing efficiency, we allow to embed the salient part of the SID in the MPLS field of the NLRI and we indicate this optimization in the SRv6 SID Structure Sub-Sub-TLV.

We defined the LISP extension in [73].

4.8.6 Implementation

From an implementation viewpoint, the integration between the overlay and the underlay represents a major challenge in the forwarding table. Several patents were filed on the subject and are key for a scalable implementation.

4.9 Service Programming and Metadata

4.9.1 Introduction

In an SR network, services, running either on a physical appliance or in a virtual environment, are associated with a segment identifier (SID). These service SIDs are then leveraged as part of a SID-list to steer packets through the corresponding services.

Service SIDs may be combined with other types of segments. SR thus provides a fully integrated solution for overlay, underlay and service programming.

Metadata for a specific service or to be exchanged between services can be encoded in the TAG field or as TLVs of the SRH.

[41] describes how a service can be associated with a SID, including legacy services with no SR capabilities, and how these service SIDs are integrated within an SR policy.

The Service Programming application leverages the SR Policy architecture, the traffic steering solution and the SR control-plane described earlier in the text. Minor control-plane extensions describing the service characteristics have been defined [75].

As opposed to the state of the art, we integrate service programming without any cost (no new protocol and hence no interworking friction) and, most-important, no state. We do so, while enabling metadata exchange between services.

Further information can be found in the selected papers: [41], [76] and [77], also available in the appendix .

4.9.2 SR-aware services

An SR-aware service can process the SR information in the packets it receives. This means being able to identify the active segment as a local instruction and move forward in the segment list, but also that the service's own behavior is not hindered due to the presence of SR information.

For example, an SR-aware firewall filtering SRv6 traffic based on its destination must retrieve that information from the last entry in the SRH rather than the Destination Address field of the IPv6 header.

An SR-aware service is associated with a locally instantiated service segment, which is used to steer traffic through it.

SR-aware services enable advanced network programming functionalities such as conditional branching and jumping to arbitrary SIDs in the segment list. In addition, SRv6 provides several ways of passing and exchanging information between services (e.g., SID arguments, tag field and metadata TLVs).

In selected paper [76] also available in the appendix, we detailed the design consideration of an SR-aware VNFs including SR encapsulation processing, recognizing inner packet, metadata processing and applying SR specific treatment to packets. We chose the firewall as an example VNF, but the design considerations have a general value that apply to several types of network functions (e.g. DPI, IDS, etc.). An SR-aware (SERA) firewall should be able to use the same set of rules defined for the legacy firewall and apply them directly to the SR encapsulated packets with no need for an SR proxy.

The implementation can apply firewall rules to inner packets of SR traffic. In addition, it can take stateless actions based on the information available in the SRH.

The implementation of SERA is open source and has been partially included in the Linux kernel (release 4.16).

4.9.3 SR-unaware services

Any service that does not meet the above criteria for SR-awareness is considered as SR-unaware.

An SR-unaware service is not able to process the SR information in the traffic that it receives. It is thus required to remove the SR information as well as any other encapsulation header before the service receives the packet, or to alter it in such a way that the service can correctly process the packet.

In [41], we define the concept of an SR proxy as an entity, separate from the service, that performs these modifications and handle the SR processing on behalf of a service. The SR proxy can run as a separate process on the service appliance, on a virtual switch or router on the compute node or on a different host.

An SR-unaware service is associated with a service segment instantiated on the SR proxy, which is used to steer traffic through the service.

Section 6 of [41] provides the pseudo-code for several SR proxy behaviors. An open source implementation experience is also reported in selected paper [77] (also available in the appendix).

Several patents have been issued on these proxy behaviors.

4.10 Scale and MTU efficiency

At MPLS congress 2019 [78], we reviewed the SRv6 deployments and highlighted that very short SID lists were needed. Hence the solution scales and performs well.

Still, one could wonder whether these properties will hold when deployments will span the Chinese or Indian sub-continents, with hundreds of sub-domains and up to 100 thousands of routers within the SP domain.

We thought about this. The solution to this problem is called “Micro-Segment”. We just submitted it to the IETF (selected paper [79], also available in the appendix) and applied for presentation at the next RIPE and NANOG conferences.

This solution is 100% consistent with the SRv6 Network Programming and SRH.

4.10.1 Intuitive idea: Program, Micro-Program, Shift and Forward

The idea is simple: as a network program is encoded as a list of SID, a SID itself could encode a micro-program as a list of micro-SIDs (called uSID).

A SID carrying a list of uSIDs is called a uSID carrier. When a uSID carrier is promoted to the outer destination address, the network routes on the uSID-block-Active-uSID prefix. Intermediate nodes on the path to the active uSID may be classic IPv6 nodes: all they see is a packet to a routed subnet.

Once the packet reaches the parent uSID node, the parent node recognizes that the active uSID is a local uN behavior. It then shifts the uSID list within the DA to the left by one uSID position. Hence the next uSID becomes active and the packet is sent on the path to its parent.

If the active uSID was the last uSID in the carrier (detected by the end of carrier), then the micro-program is ended, and the program continues: the next SID in the SRH is promoted to the outer DA.

The solution can be adapted to any type of block length and uSID length. In this note, we take the likely deployment choice: /32 block and /16 uSID.

4.10.2 Terminology

uSID carrier: a 128bit SRv6 SID of format <uSID-Block><Active-uSID><Next-uSID>...<Last-uSID><End-of-Carrier>...<End-of-Carrier>.

uSID block: A block of uSIDs

It can be any IPv6 prefix allocated to the provider (e.g. /32, /40 or /48), or it can be any block generally available for private use. An SR domain may have multiple uSID blocks.

uSID: in this document a 16-bit ID. A different length may be used.

Active uSID: first uSID after the uSID block

Next uSID: next uSID after the Active uSID

Last uSID: from left to right, the last uSID before the first End-of-Carrier uSID

End-of-Carrier: reserved ID used to mark the end of a uSID carrier. The value 0000 is selected as End-of-Carrier. All the empty uSID carrier positions must be filled with the End-of-Carrier ID. Hence, the End-of-Carrier can be present more than once in a uSID carrier.

Parent (node): the node at which an uSID is instantiated. The uSIDs are instantiated on a per-parent node basis.

Behavior of an uSID: the SRv6 function associated with a given ID.

DA[X..Y]: refers to the bits from position X to Y (included) in the IPv6 Destination Address of the received packet. The bit 0 is the MSB, while the bit 127 is the LSB.

4.10.3 uN behavior

The uN behavior is a variant of the endpoint behavior.

This behavior takes a 96b argument, "Arg", which contains the next uSIDs in the uSID carrier.

When N receives a packet whose IPv6 DA is S and S is a local uN SID, N does:

```
1.   IF DA[48..63] != 0
2.      Copy DA[48..127] into DA[32..95]
3.      Set DA[96..127] to 0x0000
4.      Forward the packet to the new DA
5.   ELSE
6.      Execute the End pseudocode
```

4.10.4 Routing

If N is configured with a uN SID BBBB:BBBB:0N00::/48 then the operator must ensure that N advertises BBBB:BBBB:0N00::/48 in routing.

4.10.5 Benefits

- Perfect integration with SRv6 Network Programming
  - SRv6 uSID is an instruction of the SRv6 network programming model
- Perfect integration with SRH
  - Any SID in DA or SRH can be an SRv6 uSID carrier
- Flexible
  - The solution can be adapted to any type of block length and uSID length
    - Here we took a BBBB:BBBB::/32 block and a /16 uSID
- Scalable SR Policy
  - 6 uSID' per uSID carrier assuming a /32 uSID block
    - As introduced earlier, an 128bit SRv6 uSID carrier is of format <uSID-Block><Active-uSID><Next-uSID>...<Last-uSID><End-of-Carrier>...<End-of-Carrier>. Assuming an uSID-Block of 32 bits (likely deployment), this leaves 128-32=96 bits filled by an ordered list of 6 uSIDs. We thus have 6 source routed waypoints per 128-bit uSID carrier.
  - 18 source routed waypoints in solely 40bytes of SRH overhead
    - T.Encaps.Red with an SRH of 40 bytes (8 fixed + 2 * 16 bytes) carries 2 extra SIDs on top of the first one in the DA.
    - With 40 bytes of SRH overhead, we thus have 3 uSID carriers
    - As explained above, we can assume 6 source routed waypoints (uSID) per carrier
    - We thus have 3*6=18 source routed waypoints in solely 40 bytes of SRH overhead
- Efficient MTU overhead
  - In apple to apple comparison, the SRv6 solution outperforms any alternative (VxLAN with SR-MPLS, CRH).
- Scalable number of globally unique nodes in the domain
  - 16-bit uSID: 65k uSIDs per domain block (*256 solely using FC/8)
  - 32-bit uSID: 4.3M uSIDs per domain block (*256 solely using FC/8)
- Hardware-friendly:
  - Leverages mature hardware capabilities (shift)
  - Avoids any extra lookup in indexed mapping table
  - Demonstrated by Cisco line-rate implementation on Broadcom Jericho1
- Control Plane friendly
  - No indexed mapping table is required
  - No routing extension is required: a simple /48 advertisement suffices
- Seamless deployment
  - SRv6 SID's in general are allocated from a different address pool than interfaces
  - SRv6 uSID's in particular are allocated from a different address pool than interfaces
  - Hence, SRv6 in general and SRv6 uSID in particular, do not require any change of physical interface addressing

5 TRAFFIC ENGINEERING

We introduce the Segment Routing Traffic Engineering solution (SR-TE) [46] in the first section. We then describe the notion of Flex-Algorithm and its usage for TE objectives. The next sections respectively describe the SR-TE architecture, the SR-TE Database, the notion of Binding SID, the scaling property of the SR-TE solution, the notion of dynamic path and SR native algorithm, the SDN control-plane and finally the tactical bandwidth optimization.

5.1 Introduction

This solution translates the intent of the operator (delay, disjointness, bandwidth) into “SR policies”, programs these SR Policies into the network, and steers traffic onto its appropriate SR Policy.

An SR Policy is fundamentally a list of segments. A list of segments, or SID-list, is represented as <S1, S2, …>, where S1 is the first segment to visit.

An SR-TE intent, defined as an optimization objective and a set of constraints, is translated into a list of segments by a compute engine.

The compute engine can be a router in which case the SDN control plane is “distributed”. It can be a Path Computation Element (PCE, referred to as SR PCE in the context of an SR deployment) in which case the SDN control plane is “centralized” or “vertical”. A network design can combine the action of the router and the SR PCE in which case the SDN control plane is called “hybrid” or “horizontal”.

The SR Policy functionality at a headend can be implemented in an SR Policy (SRP) process as illustrated in Figure 2.

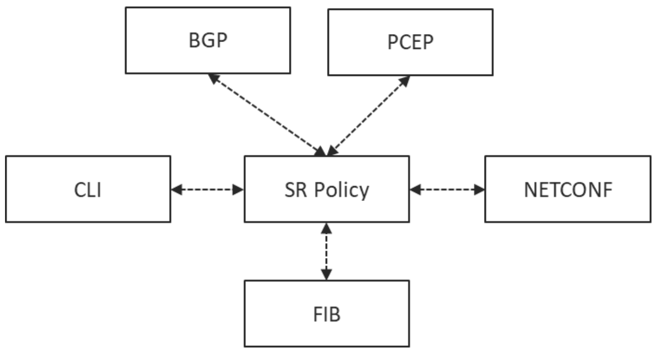


Figure 2 SR Policy Architecture at a headend

The SRP process interacts with other processes to learn candidate paths. The SRP process selects the active path of an SR Policy. The SRP process interacts with the RIB/FIB process to install an active SR Policy in the data plane.

To validate explicit candidate paths and compute dynamic candidate paths, the SRP process maintains an SR Database (SR-DB). The main source of SR-DB information is BGP-LS. Each router in the network advertises via BGP-LS its immediate topology, its services and the related SIDs.

Earlier in this document, we defined the data plane behaviors that support the SR-TE solution:

- END and END.X underlay segments
- T.Encaps(.Reduced)

In this section, we will focus on the following topics:

- Expressing a TE intent with SR-TE: with or without flexible-algorithm segments
- SR Policy architecture
- SR-TE Database
- Binding Segment
- SID list length and scaling
- Native SR Algorithms and ECMP-awareness
- Automated Traffic Engineering: ODN and AS
- SDN Control Plane
- Egress Peer Engineering
- Bandwidth Optimization

Much more could be written on the subject. We refer the reader either to [43] or the book we wrote on the subject: “Segment Routing Part II, Traffic Engineering” book we issued in May 2019 [44].

5.2 A TE intent as a SID list potentially leveraging Flex-Algo SID

There are two fundamental ways to express a TE intent with segment routing: with one single Flex-Algo SID or with multiple SIDs.

Let us start with multiple SIDs as this is likely the most intuitive.

If the shortest-path from Tokyo to Brussels is via USA (e.g. lower cost of transported bits), the low-delay path may likely be expressed with the SR Policy <toMoscow, toBrussels> as shown in Figure 3, where “toMoscow” is the first segment and represents the shortest-path from Tokyo to Moscow, and “toBrussels” is the second segment and represents the shortest-path from Moscow to Brussels.

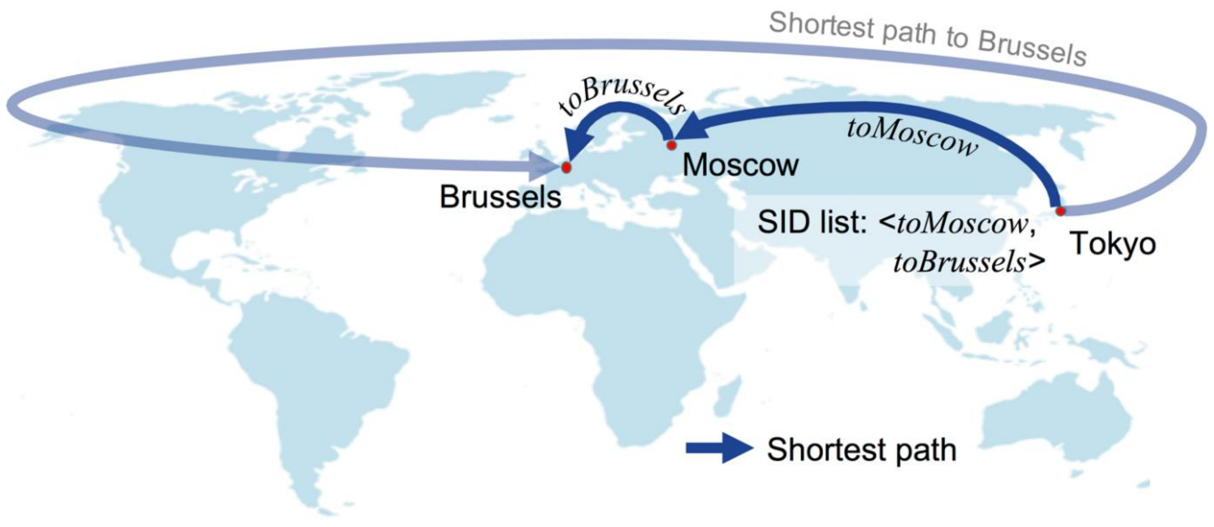


Figure 3: Low-delay path Tokyo – Brussels – using IGP shortest path SIDs

The computation engine (a router or a SR PCE) collects the topology and its segments and expresses the intent (the SR Policy) as a SID list. The salient point to remember here is that each prefix segment expresses a shortest-path according to the basic IGP metric.

There is a different way to approach the problem: defining additional segments that have specific TE properties (i.e. prefix segments that do not follow the IGP shortest-path).

Let us assume the following:

- each router monitors the propagation delay of each of its links
- the propagation delay of each link is flooded by the IGP
- each IGP node is configured with a second SID which is flooded with the "delay" attribute
- each IGP node computes the shortest-path of a "delay" SID with the per-link propagation delay as metric
- "toBrussels(Delay)" indicates the "delay" SID for Brussels

Then, it is easy to deduce that the low-delay SR policy from Tokyo to Brussels can be expressed with a single SID "toBrussels(Delay)".

Such a SID is called an "IGP Flex-Algo" SID [45] based on the following:

- IGP because the IGP computes the related shortest path (e.g. min-delay) and floods the related information inside the IGP domain
- Algo for Algorithm because we associate a specific TE intent to the SID, expressed as an optimization objective (an algorithm). Each objective is identified by an Algo number.
- Flex because any operator is free to define the intent of each Flex-Algo it instantiates
- Operator 1 may define Algo128 to minimize TE metric and exclude red affinity
- Operator 2 may define Algo128 to minimize delay metric and exclude blue affinity

The same intent (SR Policy) can thus be expressed as <toMoscow, toBrussels> or <toBrussels(Delay)> as shown in Figure 4.

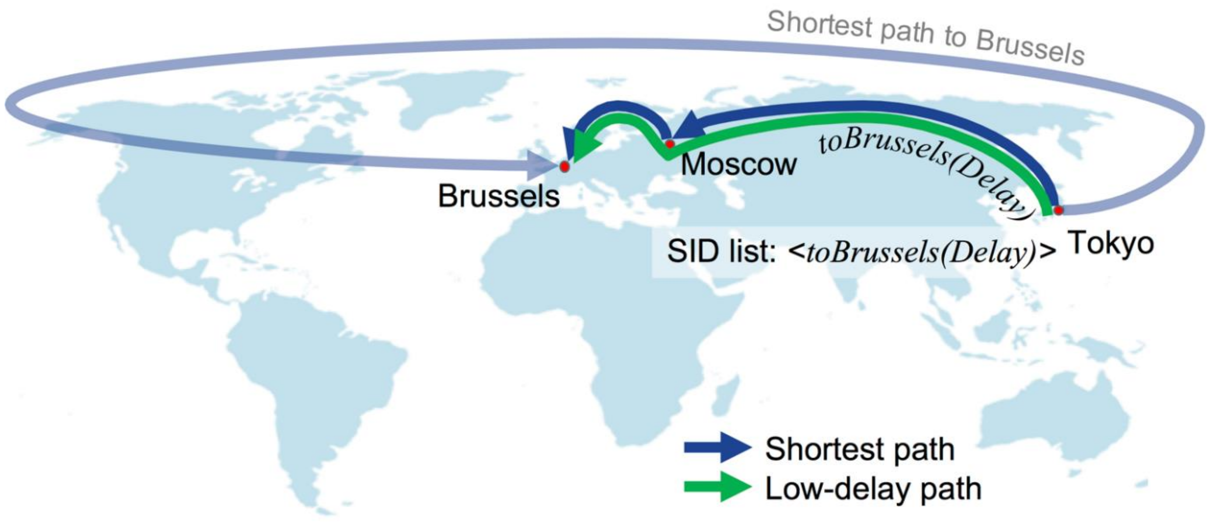


Figure 4 Low-delay path Tokyo – Brussels – using low-delay SIDs

An intent can also be expressed as a SID list of Flex-Algo SIDs. Let us assume that Moscow and Tokyo are in the Asian domain while Brussels and Moscow are in the European domain. An inter-domain low-delay path from Tokyo to Brussels could then be expressed as <toMoscow(Delay), toBrussels(Delay)>.

Similarly, an intent can be expressed as a mix of SIDs and Flex-Algo SIDs. For example, if the IGP shortest-path from Tokyo to Moscow is also the low-delay path, then the end-to-end SR policy could be expressed as <toMoscow, toBrussels(Delay)>. See Figure 5.

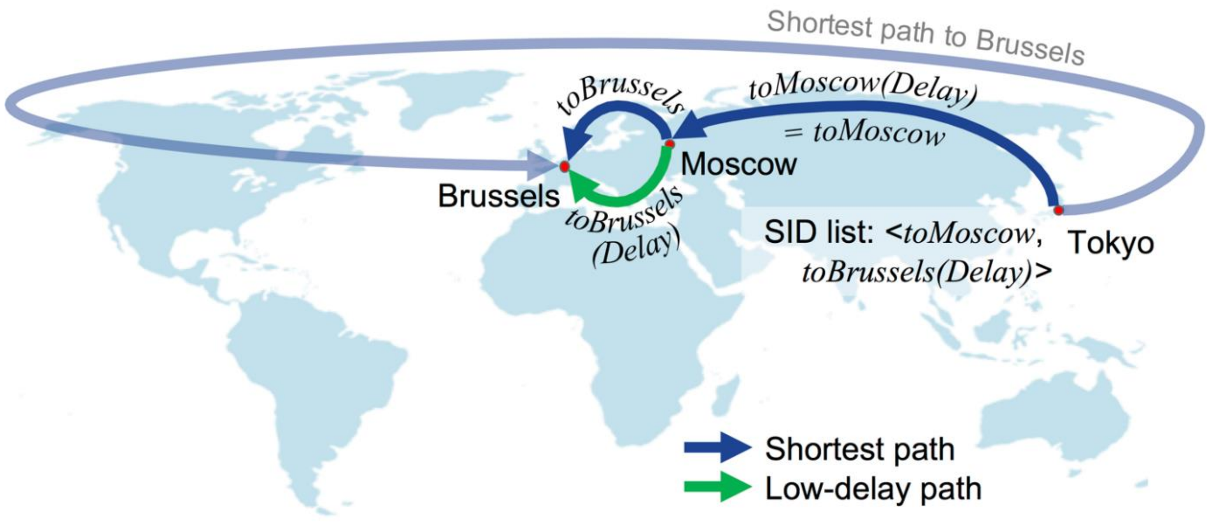

Shortest path to Brussels
toBrussels
toMoscow(Delay) = toMoscow
Moscow
Brussels
toBrussels (Delay)
SID list: <toMoscow, toBrussels(Delay)>
Tokyo
Shortest path
Low-delay path

Figure 5 Low-delay path Tokyo – Brussels – combining different algorithm SIDs

The SR-TE solution is about combining any SIDs to express the intent. If it is possible to do so with one SID, it will be done. If several SIDs are required, this is also fine. If several SIDs of a different nature are needed, this is fine as well.

The solution works like a multi-color and multi-shape construction brick system. IGP SIDs are the yellow bricks. The yellow bricks implement a minimum-IGP-cost shortest-path. Flex-Algo1 IGP SIDs are bricks of red color. The red bricks implement a minimum-delay shortest-path. Flex-Algo2 IGP SIDs are bricks of green color. The green bricks implement a minimum-IGP-cost shortest-path constrained to the green plane of the network. The blue bricks implement a minimum-IGP-cost shortest-path constrained to the blue plane of the network. Depending on the intent, the compute engine (router or SR PCE) uses different bricks. Furthermore, the solution may combine bricks of different colors (for example, as part of an inter-domain policy). This is the inherent richness of the solution.

For brevity, we will not talk more about the Flex-Algo segments in this document. The reader could refer to the IETF specification [45] and patents we wrote on the subject.

5.3 SR Policy Architecture

This section defines the SR Policy architecture. Further information can be found in [46].

An SR Policy is a framework that enables instantiation of an ordered list of segments on a node for implementing a source routing policy with a specific intent for traffic steering from that node.

The Segment Routing architecture [31] specifies that any instruction can be bound to a segment. Thus, an SR Policy can be built using any type of Segment Identifier (SID) including those associated with topological or service instructions.

In this section, we focus on the topological segments.

5.3.1 Identification of an SR Policy

An SR Policy is identified through the tuple <headend, color, endpoint>. In the context of a specific headend, one may identify an SR policy by the <color, endpoint> tuple.

The headend is the node where the policy is instantiated/implemented. The headend is specified as an IP address and is expected to be unique in the domain.

The endpoint indicates the destination of the policy. The endpoint is specified as an IP address and is expected to be unique in the domain.

The color is a 32-bit numerical value that associates the SR Policy with an intent (e.g. low-latency).

The endpoint and the color are used to automate the instantiation of SR Policies (“ODN” Cisco feature for On-Demand Next-Hop) and automate the steering of service on SR Policies (“AS” Cisco feature for Automated Steering).

The way we defined an SR Policy is critical as it directly enables a key feature that motivated most deployments: The Automated Traffic Engineering for overlay routes (ODN and AS) and the so-called Horizontal SDN model. We will detail these later in the document.

5.3.2 Candidate Path and Segment List

An SR Policy is associated with one or more candidate paths.

A candidate path is the signaling unit of an SR Policy. A headend can receive these candidate paths via configuration, Path Computation Element (PCE) Communication Protocol (PCEP) [47] or BGP SR Policy [48].

A candidate path is either dynamic or explicit.

An explicit candidate path is expressed as a Segment-List or a set of Segment-Lists.

A dynamic candidate path expresses an optimization objective and a set of constraints. The headend (potentially with the help of a PCE) computes the solution Segment-List (or set of Segment-Lists) that solves the optimization problem.

If a candidate path is associated with a set of Segment-Lists, each Segment-List is associated with a weight for weighted load balancing. The default weight is 1.

5.3.3 Preference of a Candidate Path

The preference of the candidate path is used to select the best candidate path for an SR Policy. The default preference is 100.

An operator can steer a flow over an SR Policy with two candidate paths: the preferred one when available and an alternative one when the preferred path is not available.

5.3.4 Validity of a Candidate Path

A candidate path is usable when it is valid. A common path validity criterion is the reachability of its constituent SIDs. The validation rules are specified in Section 5 of [46].

5.3.5 Active Candidate Path

A candidate path is selected when it is valid, and it is determined to be the best path of the SR Policy. The selected path is referred to as the "active path" of the SR policy in this document.

Whenever a new path is learned, or an active path is deleted, the validity of an existing path changes or an existing path is changed, the selection process MUST be re-executed.

The candidate path selection process operates on the candidate path Preference. A candidate path is selected when it is valid, and it has the highest preference value among all the candidate paths of the SR Policy.

A tie-breaking rule has been defined to deal with the case of multiple valid candidate paths of the same preference [46].

5.3.6 Validity of an SR Policy

An SR Policy is valid when it has at least one valid candidate path.

5.3.7 Instantiation of an SR Policy in the Forwarding Plane

A valid SR Policy is instantiated in the forwarding plane.

Only the active candidate path SHOULD be used for forwarding traffic that is being steered onto that policy.

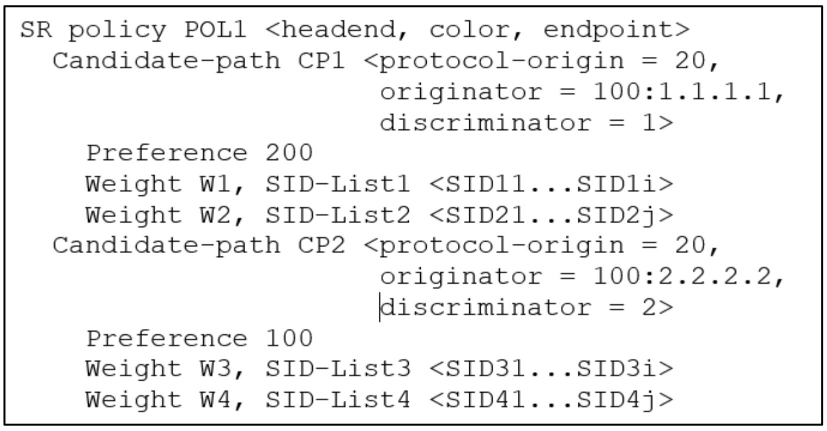
If a set of Segment-Lists is associated with the active path of the policy, then the steering is per flow and W-ECMP based according to the relative weight of each Segment-List.

The fraction of the flows associated with a given Segment-List is w/Sw where w is the weight of the Segment-List and Sw is the sum of the weights of the Segment-Lists of the selected path of the SR Policy.

The weighting among segment-lists of the active candidate path is useful to a centralized TE/SDN controller to distribute the load across the network in a fine-grain manner.

5.3.8 Summary

In summary, the information model is the following:



```
SR policy POL1 <headend, color, endpoint>
  Candidate-path CP1 <protocol-origin = 20,
                      originator = 100:1.1.1.1,
                      discriminator = 1>
    Preference 200
    Weight W1, SID-List1 <SID11...SID1i>
    Weight W2, SID-List2 <SID21...SID2j>
  Candidate-path CP2 <protocol-origin = 20,
                      originator = 100:2.2.2.2,
                      discriminator = 2>
    Preference 100
    Weight W3, SID-List3 <SID31...SID3i>
    Weight W4, SID-List4 <SID41...SID4j>
```

Figure 6 SR Policy Information Model

The SR Policy POL1 is identified by the tuple <headend, color, endpoint>. It has two candidate paths CP1 and CP2. Each is identified by a tuple <protocol-origin, originator, discriminator>. CP1 is the active candidate path (it is valid, and it has the highest preference). The two Segment-Lists of CP1 are installed as the forwarding instantiation of SR policy Pol1. Traffic steered on Pol1 is flow-based hashed on Segment-List <SID11...SID1i> with a ratio W1/(W1+W2).

5.4 Segment Routing Database

An SR headend leverages the Segment Routing Database (SR-DB) to validate explicit candidate paths and compute dynamic candidate paths.

The information in the SR-DB may include:

- IGP information (topology, IGP metrics based on ISIS and OSPF)
- Segment Routing information
- TE Link Attributes (such as TE metric, SRLG, attribute-flag, extended admin group) [49] [50]

- Extended TE Link attributes (such as latency, loss) [51] [52]
- Inter-AS Topology information [53].

The attached domain topology MAY be learned via IGP, BGP-LS or NETCONF.

A non-attached (remote) domain topology MAY be learned via BGP-LS or NETCONF.

In some use-cases, the SR-DB may only contain the attached domain topology while in others, the SR-DB may contain the topology of multiple domains and in this case, it is multi-domain capable.

5.5 Binding SID

The Binding SID (BSID) is fundamental to our solution. It provides scaling, network opacity and service independence. [54] illustrates some of these benefits.

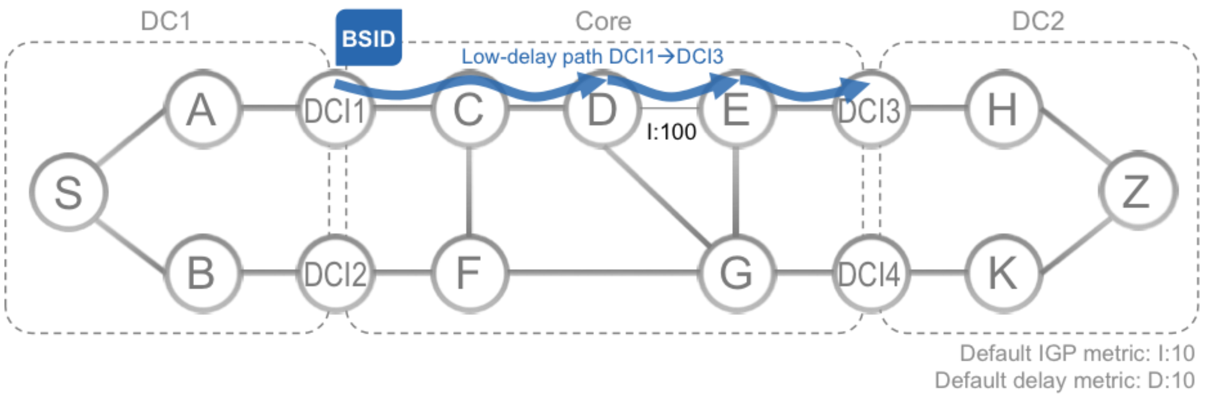


Figure 7: DC Inter-Connect with SLA and without inter-domain BGP

Let us illustrate these benefits with the diagram in Figure 7: here DCI1 has a low-delay SR Policy "Pol1" to DCI3 with SID-list <Prefix-SID(D), Adj-SID(D-to-E), Prefix-SID(DCI3)> and with a Binding SID BSID(Pol1).

In this context, a low-delay multi-domain SR Policy from S to Z is simply expressed as <Prefix-SID(DCI1), BSID(Pol1), Prefix-SID(Z)>.

Without the leverage of the intermediate core SR Policy, S would need to steer its low-delay flow into the SR Policy with SID list < Prefix-SID(DCI1), Prefix-SID(D), Adj-SID(D-to-E), Prefix-SID(DCI3), Prefix-SID(Z)>.

The use of a BSID (and the transit SR Policy) decreases the number of segments imposed by the source.

A BSID acts as a stable anchor point which isolates one domain from the churn of another domain.

Upon topology changes within the core of the network, the low-delay path from DCI1 to DCI3 may change. While the path of an intermediate policy changes, its BSID does not change. Hence the policy used by the source does not change and the source is shielded from the churn in another domain.

A BSID provides opacity and independence between domains. The administrative authority of the core domain may want to exercise a total control over the paths through this domain so that it can perform capacity planning and introduce TE for the SLAs it provides to the leaf domains. The use of a BSID allows keeping the service opaque. S is not aware of the details of how the low-delay service is provided by the core domain. S is not aware of the need of the core authority to temporarily change the intermediate path.

5.6 SID List Length and Scaling

Let us assume that a router S can only push 5 SIDs and a TE intent requires a list of 7 SIDs.

Are we stuck? No, for two reasons: Binding SID and Flex-Algo SID.

The first solution is clear as it leverages the Binding SID properties (Figure 8a).

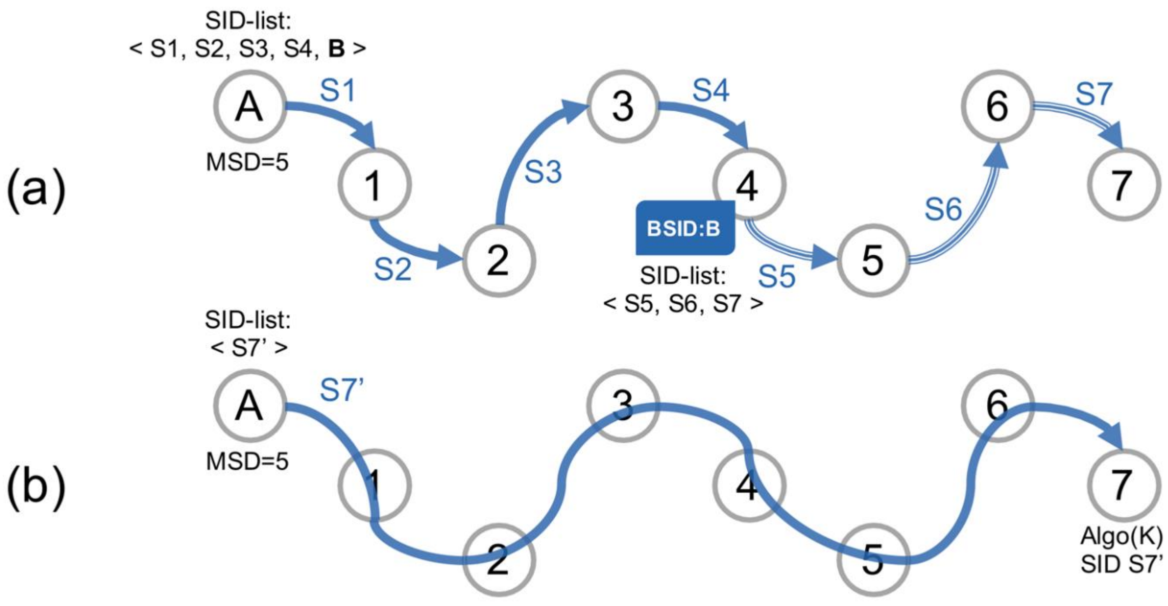


Figure 8: Solutions to handle label imposition limit

The SID list at the headend node becomes <S1, S2, S3, S4, B> where B is the Binding SID of a policy <S5, S6, S7> at node 4. The SID list at the headend node meets the 5-label constraint of that node.

While a binding SID and policy at node 4 does add state to the core for a policy on the edge, the state is not per edge policy. Therefore, a single binding SID B for policy <S5, S6, S7> may be reused by many edge node policies.

The second solution is equally straightforward: instantiate the intent on the nodes in the network as an extra Flex-Algo IGP algorithm (say AlgoK) and allocate a second SID S7' to node 7 where S7' is associated with AlgoK (Figure 8b). Clearly the SID list now becomes <S7'> and only requires one label to push.

These two concepts guarantee that an intent can be expressed as an SR Policy that meets the capabilities of the headend (e.g. max push ≤ 5 SIDs).

Outside the scope of this document, [46] details how the characteristics of the nodes are discovered (how many labels can they push) and how the optimization algorithms take this constraint into consideration.

5.7 Dynamic Path

This section describes the computation aspects of a dynamic path [54]. Specifically, for the following two optimization objectives:

- Min-Metric - requests computation of a solution Segment-List optimized for a selected metric
- Min-Metric with margin and maximum number of SIDs - Min-Metric with two changes: a margin of by which two paths with similar metrics would be considered equal, a constraint on the max number of SIDs in the Segment-List

The "Min-Metric" optimization objective requests to compute a solution Segment-List so that packets flowing through the solution Segment-List use ECMP-aware paths optimized for the selected metric.

The "Min-Metric" objective can be instantiated for the IGP metric ([55] [2] [56]) xor the TE metric ([49] [50]) xor the latency extended TE metric ([51] [52]).

This metric is called the O metric (the optimized metric) to distinguish it from the IGP metric. The solution Segment-List must be computed to minimize the number of SIDs and the number of Segment-Lists.

If the selected O metric is the IGP metric and the headend and tail-end are in the same IGP domain, then the solution Segment-List is made of the single prefix-SID of the tail-end.

When the selected O metric is not the IGP metric, then the solution Segment-List is made of prefix SIDs of intermediate nodes, Adjacency SIDs along intermediate links and potentially Binding SIDs (BSIDs) of intermediate policies.

In many deployments there are insignificant metric differences between mostly equal path (e.g. a difference of 100 usec of latency between two paths from NYC to SFO would not matter in most cases). The "Min-Metric with margin" objective supports such requirement.

The "Min-Metric with margin and maximum number of SIDs" optimization objective requests to compute a solution Segment-List such that packets flowing through the solution Segment-List do not use a path whose cumulative O metric is larger than the shortest-path O metric + margin.

If there is no solution meeting both the “Min Metric with margin” and “Maximum Number of SIDs” objectives, then one option consists in favoring the second objective (as it has a direct link to the hardware capability): i.e. to select a path with the least value of O metric which does not exceed the number of SIDs supported by the hardware. The other default option is to not come up with a solution unless the desired SLA is guaranteed.

The following constraints can be described:

- Inclusion and/or exclusion of TE affinity
- Inclusion and/or exclusion of IP address

- Inclusion and/or exclusion of SRLG
- Inclusion and/or exclusion of admin-tag
- Maximum accumulated metric (IGP, TE and latency)
- Maximum number of SIDs in the solution Segment-List
- Maximum number of weighted Segment-Lists in the solution set
- Diversity to another service instance (e.g., link, node, or SRLG disjoint paths originating from different head-ends)

These optimization objectives have been implemented in the Cisco Systems product line with SR native algorithms (see next section) and are the subject of several filed patents.

5.8 SR Native Algorithm

Let us assume in Figure 9 that all the links have the same IGP metric of 10 and let us consider the dynamic path defined as: Min-Metric(from 1, to 3, IGP metric, margin 0) with constraint "avoid link 2-to-3".

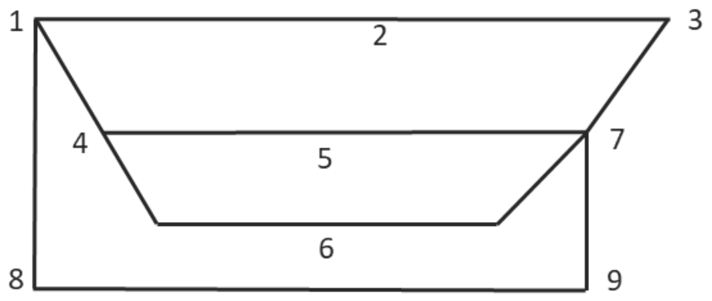


Figure 9: SR native algorithm illustration

A classical circuit implementation would do: prune the graph, compute the shortest-path, pick a single non-ECMP branch of the ECMP-aware shortest-path and encode it as a Segment-List. The solution Segment-List would be <4, 5, 7, 3>.

An SR-native algorithm would find a Segment-List that minimizes the number of SIDs and maximizes the use of all the ECMP branches along the ECMP shortest path. In this illustration, the solution Segment-List would be <7, 3>.

In the clear majority of SR use-cases, SR-native algorithms should be preferred: they preserve the native ECMP of IP and they minimize the data plane header overhead.

In some specific use-case (e.g. TDM migration over IP where the circuit notion prevails), one may prefer a classic circuit computation followed by an encoding into SIDs (potentially only using non-protected Adj SIDs that pin the path to specific links and avoid ECMP to reflect the TDM paradigm).

5.9 SDN Control Plane: Distributed and/or Centralized

5.9.1 Distributed Control Plane within a single Link-State IGP area

Consider a single-area IGP with per-link latency measurement and advertisement of the measured latency in the extended-TE IGP TLV.

A head-end H is configured with a single dynamic candidate path for SR policy P with a low-latency optimization objective and endpoint E.

Clearly the SRP process at H learns the topology (and extended TE latency information) from the IGP and computes the solution Segment-List providing the low-latency path to E.

No centralized controller is involved in such a deployment.

The SR-DB at H only uses the Link-State Data Base (LSDB) provided by the IGP.

5.9.2 Distributed Control Plane across several Link-State IGP areas

Consider a domain D composed of two link-state IGP single-area instances (I1 and I2) where each sub-domain benefits from per-link latency measurement and advertisement of the measured latency in the related IGP. The link-state information of each IGP is advertised via BGP-LS [57] towards a set of BGP-LS route reflectors (RR).

H is a headend in IGP I1 sub-domain and E is an endpoint in IGP I2 sub-domain.

Using a BGP-LS session to any BGP-LS RR, H’s SRP process may learn the link-state information of the remote domain I2. H can thus compute the low-latency path from H to E as a solution Segment-List that spans the two domains I1 and I2.

The SR-DB at H collects the LSDB from both sub-domains (I1 and I2).

No centralized controller is required.

5.9.3 Centralized Control Plane

Considering the same domain D as in the previous section, let us now assume that H does not have a BGP-LS session to the BGP-LS RRs. Instead, let us assume a controller "C" has at least one BGP-LS session to the BGP-LS RRs.

C learns the topology and extended latency information from both sub-domains via BGP-LS. It computes a low-latency path from H to E as a Segment-List <S1, S2, S3> and programs H with the related explicit candidate path.

The headend H does not compute the solution Segment-List (it cannot). It only validates the received explicit candidate path. Most probably, the controller encodes the SIDs of the Segment-List with Type-1 SID types. In that case, the headend’s validation simply consists in resolving the first SID on an outgoing interface and next-hop.

The SR-DB at H only includes the LSDB provided by the IGP I1.

The SR-DB at C includes the LSDB from both sub-domains (I1 and I2).

I often refer to this as the “Vertical SDN” model, prevalently encountered in the WEB market.

5.9.4 Distributed and Centralized Control Plane

Consider the same domain D as in the previous section.

H’s SRP process is configured to associate color C1 with a low-latency optimization objective.

H’s BGP process is configured to steer a Route R/r of extended-color community C1 and of next-hop N via an SR policy (N, C1).

Upon receiving a first BGP route of color C1 and of next-hop N, H recognizes the need for an SR Policy (N, C1) with a low-latency objective to N. As N is outside the SRTE DB of H, H requests an SR PCE to compute such Segment-List (e.g., PCEP [58]).

This is an example of hybrid control-plane: the BGP distributed control plane signals the routes and their TE requirements. Upon receiving these BGP routes, a local headend either computes the solution Segment-List (entirely distributed when the endpoint is in the SR-DB of the headend) else delegates the computation to an SR PCE (hybrid distributed/centralized control-plane).

The SR-DB at H only includes the LSDB provided by the IGP.

The SR-DB at the SR-PCE collects the LSDB from both sub-domains.

This is the “Automated Traffic Engineering solution” leveraging the ODN and AS features that we have described in the Overlay/Underlay integration section. I often refer to this as the “Horizontal SDN” model, prevalently encountered in the SP market[3].

The term horizontal helps to stress that the distributed control plane is the trigger to the SR-PCE: visually, in diagrams, the BGP updates travel from an egress PE horizontally to the left towards an ingress PE. The ingress PE then requests the SR PCE the appropriate policy.

In the vertical model, the controller is the trigger of the policy. It decides that this ingress PE needs to reach that egress PE via that path and it installs the policy independently from the eventual presence of the distributed control plane. The SR Policies are entirely governed by this central controller.

5.9.5 Flexibility and Modularity

As we explained in the beginning of this document, a primary design objective for our solution was the modularity: the abstract decomposition of complex behaviors into their basic blocks so that various deployments could combine them in different ways.

The flexibility of the SR SDN control plane illustrated in this document is an example of that modularity.

[3] The SP market refers to networks such as ATT, NTT, DT…

5.10 Tactical Bandwidth Optimization

The topic of SR-based bandwidth optimization is very broad and would need a dedicated thesis. We limit this section to a high-level overview of the work we did on the topic. Specifically, we worked on the tactical traffic engineering use-case.

In the industry, Tactical TE deployment is largely a manual process: operators are required to overprovision networks, both in terms of capacity and path diversity, to minimize the possibility of failure induced SLA violations.

In [59], we presented a method for tactical TE that allows networks to automatically mitigate failure induced SLA violations (such as link congestion) in near real time. The method is a combination of real-time traffic monitoring via the collection of the Segment Routing Demand Matrix (SRDM), a novel algorithmic optimization operating on the SRDM that computes a minimal number of "shallow" SR policies resolving the SLA violations, and the automated deployment of these tunnels via an orchestration platform.

The method has been simulated in multiple real-world customer networks and has proven to be both effective and practical. For example, in one large service provider network, Tactical SR TE resolved all failure induced SLA violations with a small (fewer that ten, often less than five) number of shallow (label stack depth at most two) SR tunnels. This continued to be true even as the simulation scaled the customer provided Demand Matrix to 140% of their current utilization (at which point the network experienced congestion even without failure) [29].

The same study showed that existing metric optimization techniques provided either no solution or only solutions requiring dozens of network changes (link metric reconfigurations) even for demands slightly exceeding 100%. The solution is therefore novel in the sense that it allows for practical network optimization where it was previously not possible.

Moreover, the simple and tactical nature of the solution makes it attractive not only to Web and Tier 1 service providers, but also to a broader base of ISPs and enterprise customers who have been reluctant to deploy complicated RSVP-TE based solutions.

Along this work, we funded and collaborated with several research teams. A noticeable outcome of one of these relationships is [60].

DEFO is a generic SR-TE path calculation tool that can combine various optimization objectives and constraints. The tool takes as input a set of traffic demands with their respective requirements in terms of end-to-end latency, bandwidth, waypoints or exclusions, and produces an appropriate Segment-List for each demand. When computing the Segment-Lists, the DEFO tool not only ensures that the returned Segment-List satisfies the requirements of its associated demand, but also prevents any conflict with other demands. For example, that the accumulated traffic steered on each link of the network never exceeds the link capacity.

To produce a result in a realistic timeframe, the DEFO tool leverages an abstraction of Segment Routing like the SR-TE metric optimization algorithm, where each segment is represented as a set of forwarding paths, combined with constraint programming techniques.

5.11 Detailed Specification

[46] is the main document to consider. It describes the SR-TE architecture and its key concepts. It introduces the various protocol extensions:

- draft-filsfils-spring-sr-traffic-counters
- draft-filsfils-spring-sr-policy-considerations
- draft-ietf-idr-bgp-ls-segment-routing-ext
- draft-ietf-idr-te-lsp-distribution
- draft-ietf-idr-bgpls-segment-routing-epe
- draft-ietf-lsr-flex-algo
- draft-ietf-pce-segment-routing
- draft-sivabalan-pce-binding-label-sid
- draft-ietf-pce-association-diversity
- draft-ietf-idr-segment-routing-te-policy
- RFC8491
- RFC8476
- draft-ietf-idr-bgp-ls-segment-routing-msd
- See the complete list on www.segment-routing.net/ietf.

6 CONCLUSION

Thanks to SRv6 Network Programming, the source controls the entire experience of the packet within the domain by encoding a stateless network program as the combination of the outer destination address (DA), the segment list and metadata in the SRH.

The functions associated with a SID may range from topological underlay or overlay instructions delivered by multi-terabit hardware engines to service-centric behaviors instantiated on CPU as entire containers or virtual machines.

These different functions can exchange data between each other thanks to the metadata TLV in the SRH.

The network programming model allows to express a wide variety of complex behaviors in an end-to-end and scalable manner through the sub-domains of the SP.

A SID may itself encode a micro-program as a list of micro-SIDs (uSID). Micro-programs guarantee scalability and efficiency of the solution even in large SP domains made of hundreds of sub-domains and hundred thousands of routers.

In this document, we described a summary of the prior art, the intuition and the solution itself.

6.1 Summary of contributions

In section 3.10, we included a comprehensive reference of our contributions (academic article, IETF standardization and patents). This contribution spans all the content of this thesis:

- Stateless-ness
- Traffic Engineering
  - ODN/AS Automation
  - Inter-Domain at scale
  - Centralized, Distributed and Hybrid control planes
- SRH with Hardware efficiency
  - SRH design, FIB Longest-Match, Shift, low MTU overhead
- 50msec Topology-Independent Prefix-Independent Fast Reroute
- Micro-Loop Avoidance
- Optimum Load Balancing
- Overlay Virtualization
- Service Programming
- Ultra Scale and Efficiency with uSID
- Operational simplicity: drastic reduction of protocols

6.2 Key insights

This section highlights some of the key insights along the SR project. This may be particularly useful to engineering students.

6.2.1 Detailed knowledge of the prior art and operational experience

In an engineering context, creativity seems very difficult without a detailed knowledge of the prior-art technology and its operational experience.

I was selected to deploy MPLS in Europe in 1998. Every month, I was in Boston learning directly from Yakov Rekhter, Dan Tappan, Eric Rosen, Bruce Davie, Bob Thomas and the MPLS founding team. This experience was invaluable, especially observing Yakov.

Then from 1998 to 2010, I worked on the biggest MPLS deployments either in terms of Traffic-Engineering, FRR, VPN or QoS. I learned the operational issues of the solution.

The first insight would be: work hard to know the prior art, ask yourself why it was built this way (people were not stupid, they were optimizing for one abstract model, you must find it else you cannot understand what really happened), absolutely get to know how it is used and what are the pain points.

For MPLS, the fundamental insight is that it was built on a replication of the ATM/FR "circuit" model. The lack of ECMP, the statefulness would then be the heart of the problem of that model. Hence, ECMP and statelessness would be the seeds of SR.

6.2.2 Centralized Traffic-Engineering

A creative insight must be validated.

As I explained in our book, the fundamental insight for the SR project was thinking of my drive from Brussels to Rome: if the shortest-path to Rome has a problem, one simply needs to tune the navigator to Chamonix and once there to Rome.

In engineering terms, the insight meant:

- using a centralized algorithm would give better determinism and optimality than the prior-art
- using of source routed policies would give ECMP and statelessness (see previous section)
- using IGP-based segments instead of OpenFlow entries would scale much better
- most important: few policies would be needed, and these policies would require few segments

The later point was "obvious" to me, but it had to be proven.

Thanks to prior projects, I had very accurate RSVP-TE data sets from the most important operators.

From a scientific viewpoint, the key moment of the project was when we simulated the first version of our centralized SR-TE algorithm and we discovered that it would beat the 50-thousand-RSVP-tunnel full-mesh of a key operator by 3 orders of magnitude: instead of operating 50k tunnels, our study showed that most congestions were resolved with 10's of SR policies and the worst-case would require less than 150 SR policies.

In the book, I described the prior art solution as “wearing raining boots and gear all the year while living in Dubai”.

This meant massive operational benefit and a completely feasible technology:

- from a hardware viewpoint, one of the key issues is the number of segments to push at an SR headend
    - our simulation solved all the problems with one single segment… just like the Brussels-to-Rome-via-Chamonix intuition
- from a control-plane viewpoint, one of the key issues is the number of policies to compute and deploy
    - our simulation showed very few policies were needed
    - anyway, we can scale the centralized compute with cloud-based techniques

We presented this in 2014 [29].

In April 2019, these results were confirmed by the Google SR deployment [5].

Thomas Telkamp, while at Global Crossings and then at Cariden, taught me a lot on analyzing data set, the art of capacity planning and the reality of RSVP-TE deployments at scale.

6.2.3 Topology Independent FRR

Once the fundamental (few policies, few segments) insight got proven, the next key technical breakthrough was TI-LFA.

I had been working on IPFRR (LFA, RLFA) for 10 years. We accumulated a lot of success, but our solution coverage was still limited to 95%-99% based on topologies. Finding a topology-independent LFA solution would finalize our work.

Obviously, with explicit routing, any path could now be expressed and hence the post-convergence path as well. We quickly realized this and started working with Pierre Francois, Ahmed Bashandy and Peter Psenak on a scalable algorithm that could be implemented on each router in a distributed manner.

In the TILFA section, we reproduced another key finding of the simulation of our algorithm to real topologies: very few segments would be required. This proved again very early that the application of source-routed policies would scale at performance on hardware.

6.2.4 µLoop Avoidance

While we had been researching on Fast Convergence for 10 years and we had major success building and deploying LFA and RLFA for the protection objective, we had no success in terms of micro-loop avoidance. We had found a few ideas that we had patented but we had not built them because we were not confident in their robustness.

When we discovered µLoop as an obvious cousin of TILFA, a third key insight came: SR would be big.

Why big? Because suddenly, quite “easily” (it still took us a few months but this is quick compared to 10 years of research), we found a solution to a day-one IP routing problem.

Furthermore, the insight of the solution was leveraging the same building blocks as TI-LFA and SR-TE: when an architecture seamlessly supports various use-cases with the same building blocks, this is an insight that things are really promising.

6.2.5 Load-Balancing

For the last 20 years, we have been organizing a yearly gathering at Cisco with key network architects from WEB, SP and Large Enterprise. As a young engineer, I loved to attend and listen. One presentation that impressed me was from, at the time, the biggest network in the world. The issue reported was lack of load-balancing accuracy and its impact on quality of service and sizing of the fiber, hence on operational cost.

I did not understand why they were making such a big point about this issue and hence I used my datasets to simulate the impact of imperfect load-balancing on capacity planning. And I understood… ECMP is key (see also 4.5)

This led to the following fundamental insight for the SR design: 1/ ECMP is key in IP networks, 2/ the non-ECMP nature of RSVP-TE circuits is a root cause of problem, 3/ SR had to be built on optimum load-balancing.

Insight 3 lead to the use of ECMP-enabled IGP segments as the foundation of the TE solution.

Insight 3 lead to a key element of the design of the SRv6 data plane solution: the ingress PE must encode entropy from the transported packet in the outer flow-label, SRv6 endpoints must use the flow-label as part of their ECMP hash. The ingress PE is anyway manipulating the transported packet and hence the computation of the hash on the inner header is straightforward. The SRv6 endpoints find the entropy of the inner packet in the outer header and hence no deep-packet inspection problem arises like in MPLS. This seems so simple and obvious that one could wonder: why highlighting this? Well, one should study the prior-art are realize how complex and suboptimum this behavior is in MPLS.

6.2.6 ODN/AS – Distributed/Hybrid Automated Traffic Engineering

While we already had found the Centralized “SDN” TE solution (6.2.2), we were still searching for a TE solution more applicable to the SP market.

We needed a distributed solution that would better match the any-to-any VPN solutions at the basis of the SP market. We needed a seamless automation of the centralized optimization in the distributed control-plane.

We described the solution in the ODN/AS, SR-PCE and hybrid control-plane sections of the thesis.

This solution has been one of the key business drivers for SR deployment.

6.2.7 Network Programming and the SRH data structure

John Leddy had been attending the yearly network architect event for many years and was seen as the key reference for IPv6. Very early in the SR project, I approached him to brainstorm on the SRv6 project and the early attempts at defining the SRH data plane.

During one of these conversations, a comment from John was key: “we should turn the network as a big computer and use segments as instructions. We should design a compiler above all of this and let application developers’ program in a much higher-level language”.

At that time, all my research had been focused on solving TE with a stateless centralized solution, providing topology-independent FRR and µLoop and the ODN/AS/SR-PCE solution. Basically, all the focus was topological/routing centered.

This comment made us realize that we could abstract a segment to a network instruction with three parts: locator, function and (local) argument. Using the C programming language as an analogy, we realized that we had to share data across instructions and hence we added the tag as a hardware-friendly cross-instruction meta-data and the TLV’s as software-compliant cross-instruction metadata. This gave us the basic data structures of the SRH. We then organized them to optimize the hardware processing: basically, the fields that the hardware would process must come earlier to avoid deep-packet inspection. Finally, still using the analogy between higher C programming language and “lower” linecard micro-code language, we realized that some instructions could “carry” micro-programs and the notion of micro-instruction flew naturally.

This led us to demonstrate together with John the first data plane application of the SRv6 network programming concept in 2016 [https://www.segment-routing.net/conferences/2016-demo-srv6-spray/]. Dave Barach was key in optimizing the header for data plane. Later on, Jisu Bhattacharya was key to implement line-rate SRv6 for our deployments on Jericho1 hardware.

6.2.8 Hardware magic sauce

Unfortunately, this is confidential and hence cannot be shared.

6.2.9 How can such a project be funded and deployed so quickly

With the aim of inspiring engineering students, we highlight some of the key business enablers for such an industrial impact.

- Brilliant executives who understood fast and empowered the team: David Ward, Ravi Chandra, Venu Venugopal back in 2013, Eyal Dagan, Jonathan Davidsson, Sumeet Ahora, Kevin Wollenberger and Vipul Deokar among many others who later helped)
- The focus on product and deployment.
- The SDN business wave was perfect timing for SR-MPLS.
- The 5G business wave is perfect for SRv6: the reachability requirement imposes IPv6. The massive investment to roll out 5G, enables IPv6 and hence enables significant technological transition (SRv6). Especially if this transition delivers more scale, more functionality and removes many protocols (cheaper operation, better robustness).

- In 2018, the Chinese government ruled that all its networks had to be IPv6 enabled: this opened a very wide avenue for SRv6 in China.
- Caring from day one for an industrial consensus and rich ecosystem
- Last but not least, working very closely with lead operators and focusing on really needed use-cases. A key metric of success for us was that all the features/solutions we productized got deployed. Avoiding boiling the ocean is key to get things done.

6.3 Future work and perspectives

In this thesis, we provided some novel solutions, based on SR, for classical network problems (such as Monitoring, Traffic Engineering and Failure Recovery). We showed that SR can provide significant enhancements with respect to other solutions and we believe that there is still room and interest for extending the achieved results in these areas. In addition, we identify and discuss a set of research directions for Segment Routing that are definitely worth exploring in the near future, which we started to have some key contributions:

6.3.1 5G

Operators are facing many challenges to operate their mobile networks because of the constant growing of traffic volume as well as the very strict latency requirements imposed by 5G. GPRS Tunneling Protocol (GTP) is the most commonly used protocol in mobile networks. GTP is used to transport users' data over service provider network. GTP has been around for a while and has some known limitations. Accordingly, the 3GPP has established a group to evaluate potential replacement for GTP.

SRv6 as is one options that 3GPP is evaluating as a stateless alternative to the GTP protocol. SRv6 integrates both the application data-path and the underlying transport layer into a single protocol, allowing operators to optimize the network in a simplified manner and removing forwarding state from the network. Examples of ongoing efforts and contributions in this direction include [83] [84] [85].

6.3.2 DC fabric

DC are used for running workloads of multi-tenants on the same physical infrastructure. Hence, they require an overlay solution to provide isolation between traffic of various tenants that need to be routed across the same fabric. While some overlay technologies such as VXLAN and NVGRE provided a solution for the multi-tenancy problem. Still, they cannot satisfy the modern Data-center requirements for service programming and Traffic engineering.

SRv6 has a great potential in DC to provide multi-tenancy, service programming and traffic engineering at scale. There are various ongoing efforts to show the potential of SRv6 in DC [86]. These efforts have led to DC deployments [87].

6.3.3 Cloud Orchestration

Another research opportunity is the integration of the SRv6 technology into Cloud orchestrators like Kubernetes (K8s). Kubernetes orchestrates and manage lifecycle of applications, deployed as containers. However, Kubernetes does not provide any solution for handling containers networking. Instead, it offloads networking to the CNI plugins. There are several CNI plugins implementations available. Still the current networking model has scalability issues specially for load balancing and service chaining.

SRv6 can provide a simple a scalable networking solution for kubernetes which solves several k8s networking challenges. There are various ongoing efforts as a collaboration between several industry leaders [88]. Part of these efforts is integrated in Contiv-VPP which is an open source network plugin for kubernetes [89].

6.3.4 NFV acceleration

As any other NFV solution, SRv6 may experience some performance limitations due to packet processing in servers' CPU. This eventually leads to higher consumption of compute resources to satisfy NFV packet forwarding requirements.

One of the solutions to this problem is offloading packet processing from server CPU to smart network interface card (NIC). Smart NICs provide a programable data plane that can be customized to perform packets processing which was typically done by server's CPU. These programable data planes allow to free the compute resource for users' applications. An example of the efforts in this direction is [90] where the SRv6 function are offloaded to the smart NIC.

6.3.5 OAM and Performance Monitoring

Any technology requires a set of OAM and performance monitoring tools that allow operators to manage and troubleshoot their network. SRv6 is no difference requires the same set of tools. While some of these tools are already available. Still, there are a lot of research opportunities in this direction. Examples of on ongoing efforts include [91] [92].

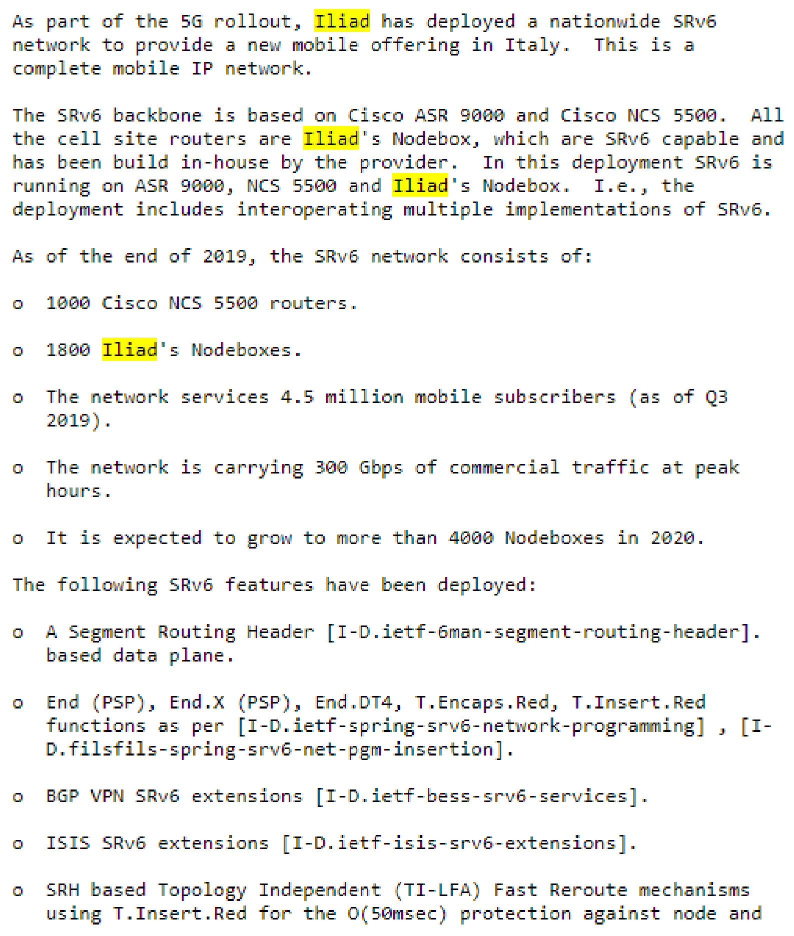
6.4 Ecosystem and Deployments

The journey along this project has been fantastic, not only from an abstract solution design but also as a tangible product development and deployment.

This section summarizes the ecosystem and deployment status.

6.4.1 Deployments

SRv6 Network Programming is deployed at Softbank and Iliad-Italy (Free) [80]. These networks have been transporting major amount of customer traffic (several hundred Gbps) for several months with linerate hardware forwarding. Figure 10 reproduces the deployment details kindly shared by Iliad-Italy [80].



```
As part of the 5G rollout, Iliad has deployed a nationwide SRv6
network to provide a new mobile offering in Italy.  This is a
complete mobile IP network.

The SRv6 backbone is based on Cisco ASR 9000 and Cisco NCS 5500.  All
the cell site routers are Iliad's Nodebox, which are SRv6 capable and
has been build in-house by the provider.  In this deployment SRv6 is
running on ASR 9000, NCS 5500 and Iliad's Nodebox.  I.e., the
deployment includes interoperating multiple implementations of SRv6.

As of the end of 2019, the SRv6 network consists of:

o  1000 Cisco NCS 5500 routers.

o  1800 Iliad's Nodeboxes.

o  The network services 4.5 million mobile subscribers (as of Q3
   2019).

o  The network is carrying 300 Gbps of commercial traffic at peak
   hours.

o  It is expected to grow to more than 4000 Nodeboxes in 2020.

The following SRv6 features have been deployed:

o  A Segment Routing Header [I-D.ietf-6man-segment-routing-header].
   based data plane.

o  End (PSP), End.X (PSP), End.DT4, T.Encaps.Red, T.Insert.Red
   functions as per [I-D.ietf-spring-srv6-network-programming] , [I-
   D.filsfils-spring-srv6-net-pgm-insertion].

o  BGP VPN SRv6 extensions [I-D.ietf-bess-srv6-services].

o  ISIS SRv6 extensions [I-D.ietf-isis-srv6-extensions].

o  SRH based Topology Independent (TI-LFA) Fast Reroute mechanisms
   using T.Insert.Red for the O(50msec) protection against node and
```

Figure 10 – Iliad Italy SRv6 Deployment

This is a noticeable achievement for a solution that has been first proposed publicly in March 2017.

6.4.2 Open-Source

The following open source platforms support SRv6 including processing of an SRH as described in [38]:

- Linux kernel 4.10, 4-14: End, End.X, End.T, End.DX2, End.DX6, End.DX4, End.DT6, End.B6, End.B6.Encaps, T.Insert, T.Encaps, T.Encaps.L2
- Linux srext module: End, End.X, End.DX2, End.DX6, End.DX4, End.AD, End.AM
- FD.io VPP: End, End.X, End.DX2, End.DX6, End.DX4, End.DT6, End.DT4, End.B6, End.B6.Encaps, End.AS, End.AD, End.AM, T.Insert, T.Encaps, T.Encaps.L2

6.4.3 Vendor

We reported to the IETF the status on vendor implementation in [80].

6.4.4 Applications

The following open-source applications have been extended to support the processing of IPv6 packets containing an SRH. For Wireshark, Tcpdump, iptables, nftables, and Snort, these extensions have been included in the mainstream version.

6.4.5 Interoperability Status of SRv6

We initiated several inter-operability events and documented them in [80] and [81].

7 BIBLIOGRAPHY

[1] D. Oran, “OSI IS-IS Intra-domain Routing Protocol,” no. 1142. RFC Editor, Feb-1990.

[2] J. Moy, “OSPF Version 2,” no. 2328. RFC Editor, Apr-1998.

[3] Y. Rekhter, S. Hares, and T. Li, “A Border Gateway Protocol 4 (BGP-4),” no. 4271. RFC Editor, Jan-2006.

[4] U. Holzle, “Open Flow @ Google,” *Open Networking Summit*, 2012. [Online]. Available: https://www.segment-routing.net/conferences/2012-ietf84-sdn-at-google/.

[5] A. Bogdanov, “Introducing Centralized TE,” *MPLS + SDN + NFV World Congress*, 2019. [Online]. Available: https://www.segment-routing.net/conferences/2019-04-11-MPLS-WC-2019-google/.

[6] C. Filsfils, K. Michielsen, and K. Talaulikar, *Segment Routing Part I*, 1st ed. USA: CreateSpace Independent Publishing Platform, 2017.

[7] A. Ebalard, “IPv6 Type 0 Routing Header,” *IETF Journal*, 2007. [Online]. Available: https://www.ietfjournal.org/ipv6-type-0-routing-header/.

[8] G. Neville-Neil, P. Savola, and J. Abley, “Deprecation of Type 0 Routing Headers in IPv6,” no. 5095. RFC Editor, Dec-2007.

[9] Y. Rekhter and E. C. Rosen, “BGP/MPLS IP Virtual Private Networks (VPNs),” no. 4364. RFC Editor, Feb-2006.

[10] M. Mahalingam *et al.*, “Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks,” no. 7348. RFC Editor, Aug-2014.

[11] P. Quinn, U. Elzur, and C. Pignataro, “Network Service Header (NSH),” no. 8300. RFC Editor, Jan-2018.

[12] Y. Zhang, “100% by 2025: China getting serious about IPv6,” *APNIC*, 2019. [Online]. Available: https://blog.apnic.net/2019/06/06/100-by-2025-china-getting-serious-about-ipv6/.

[13] J. M. Batalla, G. Mastorakis, C. X. Mavromoustakis, C. Dobre, N. Chilamkurti, and S. Schaeckeler, “Network Services Chaining in the 5G Vision,” *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 112–113, Nov. 2017.

[14] J. Evans and C. Filsfils, *Deploying IP and MPLS QoS for Multiservice Networks: Theory & Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007.

[15] C. Filsfils and J. Evans, “Engineering a Multiservice IP Backbone to Support Tight SLAs,” *Comput. Netw.*, vol. 40, no. 1, pp. 131–148, Sep. 2002.

[16] J. Evans and C. Filsfils, “Deploying Diffserv at the network edge for tight SLAs, Part 2,” *IEEE Internet Comput.*, vol. 8, no. 2, pp. 61–69, Mar. 2004.

[17] J. Evans and C. Filsfils, “Deploying Diffserv at the Network Edge for Tight SLAs, Part 1,” *IEEE Internet Comput.*, vol. 8, no. 1, pp. 61–65, Jan. 2004.

[18] C. Filsfils and J. Evans, “Deploying diffserv in backbone networks for tight SLA control,” *IEEE*

Internet Comput., vol. 9, no. 1, pp. 66–74, Jan. 2005.

[19] O. Gerstel *et al.*, "Multi-layer capacity planning for IP-optical networks," *IEEE Commun. Mag.*, vol. 52, no. 1, pp. 44–51, Jan. 2014.

[20] J. Evans, A. Begen, J. Greengrass, and C. Filsfils, "Toward Lossless Video Transport," *IEEE Internet Comput.*, vol. 15, no. 6, pp. 48–57, Nov. 2011.

[21] C. Filsfils *et al.*, "Loop-Free Alternate (LFA) Applicability in Service Provider (SP) Networks," no. 6571. RFC Editor, Jun-2012.

[22] S. Bryant, C. Filsfils, S. Previdi, M. Shand, and N. So, "Remote Loop-Free Alternate (LFA) Fast Reroute (FRR)," no. 7490. RFC Editor, Apr-2015.

[23] P. Francois, C. Filsfils, J. Evans, and O. Bonaventure, "Achieving Sub-second IGP Convergence in Large IP Networks," *SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 3, pp. 35–44, Jul. 2005.

[24] O. Bonaventure, C. Filsfils, and P. Francois, "Achieving Sub-50 Milliseconds Recovery Upon BGP Peering Link Failures," *IEEE/ACM Trans. Netw.*, vol. 15, no. 5, pp. 1123–1135, Oct. 2007.

[25] O. Bonaventure, C. Filsfils, and P. Francois, "Achieving Sub-50 Milliseconds Recovery Upon BGP Peering Link Failures," in *Proceedings of the 2005 ACM Conference on Emerging Network Experiment and Technology*, 2005, pp. 31–42.

[26] C. Hong *et al.*, "Achieving High Utilization with Software-driven WAN," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013, pp. 15–26.

[27] S. Jain *et al.*, "B4: Experience with a Globally-deployed Software Defined Wan," in *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, 2013, pp. 3–14.

[28] P. Mattes, "Segment Routing for Datacenter Interconnect at Scale," *MPLS + SDN + NFV World Congress*, 2017. [Online]. Available: https://www.segment-routing.net/conferences/2017-mpls-world-congress-paul-mattes/.

[29] C. Filsfils, "Tactical Centralized SR-based Traffic Enginering," *MPLS + SDN + NFV World Congress*, 2014. [Online]. Available: https://www.segment-routing.net/conferences/2014-mpls-sdn-2014-tactical-centralized-sr-based-traffic-enginering/.

[30] C. Filsfils, P. Camarillo, J. Leddy, D. Voyer, S. Matsushima, and Z. Li, "SRv6 Network Programming," no. draft-ietf-spring-srv6-network-programming-00. Apr-2019.

[31] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment Routing Architecture," no. 8402. RFC Editor, Jul-2018.

[32] C. Filsfils, S. F. Bryant, and D. C. Frost, "Segment Routing Techniques," *U.S. Patent 20140169370A1*, 2014. [Online]. Available: https://patents.google.com/patent/US20140169370.

[33] C. Filsfils, N. K. Nainar, C. Pignataro, J. C. Cardona, and P. Francois, "The Segment Routing Architecture," in *2015 IEEE Global Communications Conference (GLOBECOM)*, 2015, pp. 1–6.

[34] C. Filsfils, F. Clad, P. Camarillo, and D. Ward, "Enhanced Segment Routing Processing of Packets," *U.S. Patent 20180375766A1*, 2018. [Online]. Available: https://patents.google.com/patent/US20180375766A1.

[35] P. Psenak, C. Filsfils, A. Bashandy, B. Decraene, and Z. Hu, “IS-IS Extension to Support Segment Routing over IPv6 Dataplane,” no. draft-ietf-lsr-isis-srv6-extensions-02. Internet Engineering Task Force, Jul-2019.

[36] G. Dawra, C. Filsfils, K. Talaulikar, M. Chen, D. Bernier, and B. Decraene, “BGP Link State Extensions for SRv6,” no. draft-ietf-idr-bgpls-srv6-ext-01. Internet Engineering Task Force, Jul-2019.

[37] G. Dawra *et al.*, “SRv6 BGP based Overlay services,” no. draft-dawra-bess-srv6-services-00. Internet Engineering Task Force, Mar-2019.

[38] C. Filsfils, D. Dukes, S. Previdi, J. Leddy, S. Matsushima, and D. Voyer, “IPv6 Segment Routing Header (SRH),” *Internet Engineering Task Force*, no. draft-ietf-6man-segment-routing-header-21. Internet Engineering Task Force, Jun-2019.

[39] S. Previdi and C. Filsfils, “Segment routing extension headers,” *U.S. Patent US9762488B2*, 2017. [Online]. Available: https://patents.google.com/patent/US9762488B2.

[40] S. Deering and B. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” no. 8200. RFC Editor, Jul-2017.

[41] F. Clad *et al.*, “Service Programming with Segment Routing,” no. draft-xuclad-spring-sr-service-programming-02. Internet Engineering Task Force, Apr-2019.

[42] S. Amante and B. Carpenter, “Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels,” no. 6438. RFC Editor, Nov-2011.

[43] L. Ginsberg, S. Previdi, Q. Wu, J. Tantsura, and C. Filsfils, “BGP - Link State (BGP-LS) Advertisement of IGP Traffic Engineering Performance Metric Extensions,” no. 8571. RFC Editor, Mar-2019.

[44] C. Filsfils, K. Michielsen, and F. Clad, *Segment Routing Part II: Traffic Engineering*, 2nd ed. Independently published, 2019.

[45] P. Psenak, S. Hegde, C. Filsfils, K. Talaulikar, and A. Gulko, “IGP Flexible Algorithm,” no. draft-ietf-lsr-flex-algo-03. Internet Engineering Task Force, Jul-2019.

[46] C. Filsfils, S. Sivabalan, D. Voyer, A. Bogdanov, and P. Mattes, “Segment Routing Policy Architecture,” no. draft-ietf-spring-segment-routing-policy-03. Internet Engineering Task Force, May-2019.

[47] E. Crabbe, I. Minei, S. Sivabalan, and R. Varga, “Path Computation Element Communication Protocol (PCEP) Extensions for PCE-Initiated LSP Setup in a Stateful PCE Model,” no. 8281. RFC Editor, Dec-2017.

[48] S. Previdi, C. Filsfils, P. Mattes, E. C. Rosen, D. Jain, and S. Lin, “Advertising Segment Routing Policies in BGP,” no. draft-ietf-idr-segment-routing-te-policy-07. Internet Engineering Task Force, Jul-2019.

[49] T. Li and H. Smit, “IS-IS Extensions for Traffic Engineering,” no. 5305. RFC Editor, Oct-2008.

[50] D. M. Yeung, D. Katz, and K. Kompella, “Traffic Engineering (TE) Extensions to OSPF Version 2,” no. 3630. RFC Editor, Oct-2003.

[51] L. Ginsberg, S. Previdi, S. Giacalone, D. Ward, J. Drake, and Q. Wu, “IS-IS Traffic Engineering (TE)

Metric Extensions," no. 8570. RFC Editor, Mar-2019.

[52] S. Giacalone, D. Ward, J. Drake, A. Atlas, and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions," no. 7471. RFC Editor, Mar-2015.

[53] S. Previdi, K. Talaulikar, C. Filsfils, K. Patel, S. Ray, and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering," no. draft-ietf-idr-bgpls-segment-routing-epe-19. Internet Engineering Task Force, May-2019.

[54] C. Filsfils, K. Talaulikar, P. G. Król, M. Horneffer, and P. Mattes, "SR Policy Implementation and Deployment Considerations," no. draft-filsfils-spring-sr-policy-considerations-03. Internet Engineering Task Force, Apr-2019.

[55] "Use of OSI IS-IS for routing in TCP/IP and dual environments," no. 1195. RFC Editor, Dec-1990.

[56] D. Ferguson, A. Lindem, and J. Moy, "OSPF for IPv6," no. 5340. RFC Editor, Jul-2008.

[57] H. Gredler, J. Medved, S. Previdi, A. Farrel, and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP," no. 7752. RFC Editor, Mar-2016.

[58] S. Sivabalan, C. Filsfils, J. Tantsura, W. Henderickx, and J. Hardwick, "PCEP Extensions for Segment Routing," no. draft-ietf-pce-segment-routing-16. Internet Engineering Task Force, Mar-2019.

[59] T. I. M. LaBERGE, C. Filsfils, and P. FRANCOIS, "Tactical traffic engineering based on segment routing policies," *U.S. Patent US10212088B2*, 2019. [Online]. Available: https://patents.google.com/patent/US10212088B2.

[60] R. Hartert *et al.*, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, 2015, pp. 15–28.

[61] S. Litkowski *et al.*, "Topology Independent Fast Reroute using Segment Routing," no. draft-ietf-rtgwg-segment-routing-ti-lfa-01. Internet Engineering Task Force, Mar-2019.

[62] P. Sarkar, S. Litkowski, B. Decraene, C. Filsfils, K. Raza, and M. Horneffer, "Operational Management of Loop-Free Alternates," no. 7916. RFC Editor, Jul-2016.

[63] S. Litkowski, B. Decraene, C. Filsfils, and P. Francois, "Micro-loop Prevention by Introducing a Local Convergence Delay," no. 8333. RFC Editor, Mar-2018.

[64] A. Bashandy, C. Filsfils, S. Litkowski, P. Francois, and P. Psenak, "Loop avoidance using Segment Routing," no. draft-bashandy-rtgwg-segment-routing-uloop-06. Internet Engineering Task Force, Jul-2019.

[65] C. Filsfils, B. Duvivier, and M. Sivabalan, "On-demand next-hop resolution," *U.S. Patent US20170230274A1*, 2017. [Online]. Available: https://patents.google.com/patent/US20170230274A1.

[66] P. Brissette, C. Filsfils, D. Dukes, G. Dawra, F. Clad, and P. Camarillo, "Ethernet Virtual Private Network (EVPN) using an Internet Protocol Version 6 Segment Routing (SRv6) Underlay Network and SRv6-enhanced Border Gateway Protocol (BGP) Signaling," *U.S. Patent US20180375763A1*, 2018. [Online]. Available: https://patents.google.com/patent/US20180375763A1.

[67] D. Lebrun, M. Jadin, F. Clad, C. Filsfils, and O. Bonaventure, “Software Resolved Networks: Rethinking Enterprise Networks with IPv6 Segment Routing,” in *Proceedings of the Symposium on {SDN} Research, {SOSR} 2018, Los Angeles, CA, USA, March 28-29, 2018*, 2018, pp. 6:1--6:14.

[68] S. Previdi, C. Filsfils, B. Decraene, S. Litkowski, M. Horneffer, and R. Shakir, “Source Packet Routing in Networking (SPRING) Problem Statement and Requirements,” no. 7855. RFC Editor, May-2016.

[69] C. Filsfils, S. Previdi, G. Dawra, E. Aries, and D. Afanasiev, “Segment Routing Centralized BGP Egress Peer Engineering,” no. draft-ietf-spring-segment-routing-central-epe-10. Internet Engineering Task Force, Dec-2017.

[70] B. Schlinker *et al.*, “Engineering Egress with Edge Fabric: Steering Oceans of Content to the World,” in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 418–431.

[71] K. Yap *et al.*, “Taking the Edge off with Espresso: Scale, Reliability and Programmability for Global Internet Peering,” in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 432–445.

[72] D. Dukes *et al.*, “SR For SDWAN: VPN with Underlay SLA,” no. draft-dukes-spring-sr-for-sdwan-02. Internet Engineering Task Force, Jun-2019.

[73] A. Rodriguez-Natal, V. Ermagan, F. Maino, D. Dukes, P. C. Garvia, and C. Filsfils, “LISP Control Plane for SRv6 Endpoint Mobility,” no. draft-rodrigueznatal-lisp-srv6-02. Internet Engineering Task Force, Jul-2019.

[74] S. Previdi, C. Filsfils, A. Lindem, A. Sreekantiah, and H. Gredler, “Segment Routing Prefix SID extensions for BGP,” no. draft-ietf-idr-bgp-prefix-sid-27. Internet Engineering Task Force, Jun-2018.

[75] G. Dawra *et al.*, “BGP-LS Advertisement of Segment Routing Service Segments,” no. draft-dawra-idr-bgp-ls-sr-service-segments-02. Internet Engineering Task Force, Jul-2019.

[76] A. Abdelsalam, S. Salsano, F. Clad, P. Camarillo, and C. Filsfils, “SERA: SEgment Routing Aware Firewall for Service Function Chaining scenarios,” in *2018 IFIP Networking Conference (IFIP Networking) and Workshops*, 2018, pp. 46–54.

[77] A. Abdelsalam, F. Clad, C. Filsfils, S. Salsano, G. Siracusano, and L. Veltri, “Implementation of virtual network function chaining through segment routing in a linux-based NFV infrastructure,” in *2017 IEEE Conference on Network Softwarization (NetSoft)*, 2017, pp. 1–5.

[78] C. Filsfils, “SR Deployment Experience and Technology Update,” *MPLS + SDN + NFV World Congress*, 2019. .

[79] C. Filsfils *et al.*, “Network Programming extension: SRv6 uSID instruction,” no. draft-filsfils-spring-net-pgm-extension-srv6-usid-00. Internet Engineering Task Force, Jul-2019.

[80] S. Matsushima, C. Filsfils, Z. Ali, and Z. Li, “SRv6 Implementation and Deployment Status,” no. draft-matsushima-spring-srv6-deployment-status-01. Internet Engin, May-2019.

[81] C. Filsfils *et al.*, “SRv6 interoperability report,” no. draft-filsfils-spring-srv6-interop-02. Internet Engineering Task Force, Mar-2019.

[82] Awduche et al. "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC3209, December 2001

[83] S. Matsushima *et al.*, “Segment Routing IPv6 for Mobile User Plane,” no. draft-ietf-dmm-srv6-mobile-uplane-07. Internet Engineering Task Force, Nov-2019.

[84] J. Horn , “5G dataplane using SRv6,” 2018. [Online]. Available: https://www.youtube.com/watch?v=naph379CtdY.

[85] S. Matsushima, “SRv6 for 5G Mobile Update,” JANOG43 Meeting, 2019. [Online]. Available: https://www.janog.gr.jp/meeting/janog43/application/files/4315/4820/8008/janog43-matsushima-srv6.pdf.

[86] Cisco blogs, “Cisco Introduces Segment Routing v6 on Nexus 9000 GX Series Platforms,” 2020. [Online]. Available: https://blogs.cisco.com/datacenter/cisco-introduces-segment-routing-v6-on-nexus-9000-gx-series-platforms.

[87] Hirofumi Ichihara et al, “LINE Data Center Networking with SRv6,” 2019. [Online]. Available: https://www.segment-routing.net/conferences/2019-09-20-SRv6-LINE-DC/.

[88] A. Abdelsalam, Miroslaw Walukiewicz, Filip Gschwandtner, Daniel Bernier, “Rethinking kubernetes networking with SRv6 and Contiv-VPP,” FOSDEM, 2020. [Online]. Available: https://fosdem.org/2020/schedule/event/rethinking_kubernetes_networking_with_srv6/.

[89] R. Szabo, “SRv6 (Segment Routing on IPv6) Implementation of K8s Services,” 2019. [Online]. Available: https://github.com/contiv/vpp/blob/master/docs/setup/SRV6.md.

[90] HCL Technologies, “Segment Routing Over IPv6 Acceleration Using Intel FPGA Programmable Acceleration Card N3000,” 2019. [Online]. Available: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01295-hcl-segment-routing-over-ipv6-acceleration-using-intel-fpga-programmable-acceleration-card-n3000.pdf.

[91] R. Gandhi *et al.*, “Performance Measurement Using TWAMP Light for Segment Routing Networks,” no. draft-gandhi-spring-twamp-srpm-08. Internet Engineering Task Force, Mar-2020.

[92] R. Gandhi *et al.*, “Performance Measurement Using STAMP for Segment Routing Networks,” no. draft-gandhi-spring-stamp-srpm-00. Internet Engineering Task Force, Mar-2020.

[93] P. L. Ventre, S. Salsano, M. Polverini, A. Cianfrani, A. Abdelsalam, C. Filsfils, P. Camarillo, and F. Clad, “Segment Routing: a Comprehensive Survey of Research Activities, Standardization Efforts and Implementation Results,” submitted paper under a second review round in IEEE surveys and Tutorials, arXiv preprint arXiv:1904.03471v2, 2020.

Internet Engineering Task Force (IETF)
Request for Comments: 8402
Category: Standards Track
ISSN: 2070-1721

C. Filsfils, Ed.
S. Previdi, Ed.
L. Ginsberg
Cisco Systems, Inc.
B. Decraene
S. Litkowski
Orange
R. Shakir
Google, Inc.
July 2018

Segment Routing Architecture

Abstract

Segment Routing (SR) leverages the source routing paradigm. A node steers a packet through an ordered list of instructions, called "segments". A segment can represent any instruction, topological or service based. A segment can have a semantic local to an SR node or global within an SR domain. SR provides a mechanism that allows a flow to be restricted to a specific topological path, while maintaining per-flow state only at the ingress node(s) to the SR domain.

SR can be directly applied to the MPLS architecture with no change to the forwarding plane. A segment is encoded as an MPLS label. An ordered list of segments is encoded as a stack of labels. The segment to process is on the top of the stack. Upon completion of a segment, the related label is popped from the stack.

SR can be applied to the IPv6 architecture, with a new type of routing header. A segment is encoded as an IPv6 address. An ordered list of segments is encoded as an ordered list of IPv6 addresses in the routing header. The active segment is indicated by the Destination Address (DA) of the packet. The next active segment is indicated by a pointer in the new routing header.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Segment Routing (SR) leverages the source routing paradigm. A node steers a packet through an SR Policy instantiated as an ordered list of instructions called "segments". A segment can represent any instruction, topological or service based. A segment can have a semantic local to an SR node or global within an SR domain. SR supports per-flow explicit routing while maintaining per-flow state only at the ingress nodes to the SR domain.

A segment is often referred to by its Segment Identifier (SID).

A segment may be associated with a topological instruction. A topological local segment may instruct a node to forward the packet via a specific outgoing interface. A topological global segment may instruct an SR domain to forward the packet via a specific path to a destination. Different segments may exist for the same destination, each with different path objectives (e.g., which metric is minimized, what constraints are specified).

A segment may be associated with a service instruction (e.g., the packet should be processed by a container or Virtual Machine (VM) associated with the segment). A segment may be associated with a QoS treatment (e.g., shape the packets received with this segment at x Mbps).

The SR architecture supports any type of instruction associated with a segment.

The SR architecture supports any type of control plane: distributed, centralized, or hybrid.

In a distributed scenario, the segments are allocated and signaled by IS-IS or OSPF or BGP. A node individually decides to steer packets on an SR Policy (e.g., pre-computed local protection [RFC8355]). A node individually computes the SR Policy.

In a centralized scenario, the segments are allocated and instantiated by an SR controller. The SR controller decides which nodes need to steer which packets on which source-routed policies. The SR controller computes the source-routed policies. The SR architecture does not restrict how the controller programs the network. Likely options are Network Configuration Protocol (NETCONF), Path Computation Element Communication Protocol (PCEP), and BGP. The SR architecture does not restrict the number of SR controllers. Specifically, multiple SR controllers may program the same SR domain. The SR architecture allows these SR controllers to discover which SIDs are instantiated at which nodes and which sets of local (SRLB) and global (SRGB) labels are available at which node.

A hybrid scenario complements a base distributed control plane with a centralized controller. For example, when the destination is outside the IGP domain, the SR controller may compute an SR Policy on behalf of an IGP node. The SR architecture does not restrict how the nodes that are part of the distributed control plane interact with the SR controller. Likely options are PCEP and BGP.

Hosts MAY be part of an SR domain. A centralized controller can inform hosts about policies either by pushing these policies to hosts or by responding to requests from hosts.

The SR architecture can be instantiated on various data planes. This document introduces two data-plane instantiations of SR: SR over MPLS (SR-MPLS) and SR over IPv6 (SRv6).

SR can be directly applied to the MPLS architecture with no change to the forwarding plane [SR-MPLS]. A segment is encoded as an MPLS label. An SR Policy is instantiated as a stack of labels. The segment to process (the active segment) is on the top of the stack. Upon completion of a segment, the related label is popped from the stack.

SR can be applied to the IPv6 architecture with a new type of routing header called the SR Header (SRH) [IPv6-SRH]. An instruction is associated with a segment and encoded as an IPv6 address. An SRv6 segment is also called an SRv6 SID. An SR Policy is instantiated as an ordered list of SRv6 SIDs in the routing header. The active segment is indicated by the Destination Address (DA) of the packet. The next active segment is indicated by the SegmentsLeft (SL) pointer in the SRH. When an SRv6 SID is completed, the SL is decremented and the next segment is copied to the DA. When a packet is steered on an SR Policy, the related SRH is added to the packet.

In the context of an IGP-based distributed control plane, two topological segments are defined: the IGP-Adjacency segment and the IGP-Prefix segment.

In the context of a BGP-based distributed control plane, two topological segments are defined: the BGP peering segment and the BGP-Prefix segment.

The headend of an SR Policy binds a SID (called a Binding segment or BSID) to its policy. When the headend receives a packet with active segment matching the BSID of a local SR Policy, the headend steers the packet into the associated SR Policy.

This document defines the IGP, BGP, and Binding segments for the SR-MPLS and SRv6 data planes.

Note: This document defines the architecture for Segment Routing, including definitions of basic objects and functions and a description of the overall design. It does NOT define the means of implementing the architecture -- that is contained in numerous referenced documents, some of which are mentioned in this document as a convenience to the reader.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

SR-MPLS: the instantiation of SR on the MPLS data plane.

SRv6: the instantiation of SR on the IPv6 data plane.

Segment: an instruction a node executes on the incoming packet (e.g., forward packet according to shortest path to destination, or, forward packet through a specific interface, or, deliver the packet to a given application/service instance).

SID: a segment identifier. Note that the term SID is commonly used in place of the term "Segment", though this is technically imprecise as it overlooks any necessary translation.

SR-MPLS SID: an MPLS label or an index value into an MPLS label space explicitly associated with the segment.

SRv6 SID: an IPv6 address explicitly associated with the segment.

Segment Routing domain (SR domain): the set of nodes participating in the source-based routing model. These nodes may be connected to the same physical infrastructure (e.g., a Service Provider's network). They may as well be remotely connected to each other (e.g., an enterprise VPN or an overlay). If multiple protocol instances are deployed, the SR domain most commonly includes all of the protocol instances in a network. However, some deployments may wish to subdivide the network into multiple SR domains, each of which includes one or more protocol instances. It is expected that all nodes in an SR domain are managed by the same administrative entity.

Active Segment: the segment that is used by the receiving router to process the packet. In the MPLS data plane, it is the top label. In the IPv6 data plane, it is the destination address [IPv6-SRH].

PUSH: the operation consisting of the insertion of a segment at the top of the segment list. In SR-MPLS, the top of the segment list is the topmost (outer) label of the label stack. In SRv6, the top of the segment list is represented by the first segment in the Segment Routing Header as defined in [IPv6-SRH].

NEXT: when the active segment is completed, NEXT is the operation consisting of the inspection of the next segment. The next segment becomes active. In SR-MPLS, NEXT is implemented as a POP of the top label. In SRv6, NEXT is implemented as the copy of the next segment from the SRH to the destination address of the IPv6 header.

CONTINUE: the active segment is not completed; hence, it remains active. In SR-MPLS, the CONTINUE operation is implemented as a SWAP of the top label [RFC3031]. In SRv6, this is the plain IPv6 forwarding action of a regular IPv6 packet according to its destination address.

SR Global Block (SRGB): the set of global segments in the SR domain. If a node participates in multiple SR domains, there is one SRGB for each SR domain. In SR-MPLS, SRGB is a local property of a node and identifies the set of local labels reserved for global segments. In SR-MPLS, using identical SRGBs on all nodes within the SR domain is strongly recommended. Doing so eases operations and troubleshooting as the same label represents the same global segment at each node. In SRv6, the SRGB is the set of global SRv6 SIDs in the SR domain.

SR Local Block (SRLB): local property of an SR node. If a node participates in multiple SR domains, there is one SRLB for each SR domain. In SR-MPLS, SRLB is a set of local labels reserved for local segments. In SRv6, SRLB is a set of local IPv6 addresses reserved for local SRv6 SIDs. In a controller-driven network, some controllers or applications may use the control plane to discover the available set of local segments.

Global Segment: a segment that is part of the SRGB of the domain. The instruction associated with the segment is defined at the SR domain level. A topological shortest-path segment to a given destination within an SR domain is a typical example of a global segment.

Local Segment: In SR-MPLS, this is a local label outside the SRGB. It may be part of the explicitly advertised SRLB. In SRv6, this can be any IPv6 address, i.e., the address may be part of the SRGB, but used such that it has local significance. The instruction associated with the segment is defined at the node level.

IGP Segment: the generic name for a segment attached to a piece of information advertised by a link-state IGP, e.g., an IGP prefix or an IGP adjacency.

IGP-Prefix Segment: an IGP-Prefix segment is an IGP segment representing an IGP prefix. When an IGP-Prefix segment is global within the SR IGP instance/topology, it identifies an instruction to forward the packet along the path computed using the routing algorithm specified in the algorithm field, in the topology, and in the IGP instance where it is advertised. Also referred to as "prefix segment".

Prefix-SID: the SID of the IGP-Prefix segment.

IGP-Anycast Segment: an IGP-Anycast segment is an IGP-Prefix segment that identifies an anycast prefix advertised by a set of routers.

Anycast-SID: the SID of the IGP-Anycast segment.

IGP-Adjacency Segment: an IGP-Adjacency segment is an IGP segment attached to a unidirectional adjacency or a set of unidirectional adjacencies. By default, an IGP-Adjacency segment is local (unless explicitly advertised otherwise) to the node that advertises it. Also referred to as "Adj-SID".

Adj-SID: the SID of the IGP-Adjacency segment.

IGP-Node Segment: an IGP-Node segment is an IGP-Prefix segment that identifies a specific router (e.g., a loopback). Also referred to as "Node Segment".

Node-SID: the SID of the IGP-Node segment.

SR Policy: an ordered list of segments. The headend of an SR Policy steers packets onto the SR Policy. The list of segments can be specified explicitly in SR-MPLS as a stack of labels and in SRv6 as an ordered list of SRv6 SIDs. Alternatively, the list of segments is computed based on a destination and a set of optimization objective and constraints (e.g., latency, affinity, SRLG, etc.). The computation can be local or delegated to a PCE server. An SR Policy can be configured by the operator, provisioned via NETCONF [RFC6241] or provisioned via PCEP [RFC5440]. An SR Policy can be used for Traffic Engineering (TE), Operations, Administration, and Maintenance (OAM), or Fast Reroute (FRR) reasons.

Segment List Depth: the number of segments of an SR Policy. The entity instantiating an SR Policy at a node N should be able to discover the depth-insertion capability of the node N. For example, the PCEP SR capability advertisement described in [PCEP-SR-EXT] is one means of discovering this capability.

Forwarding Information Base (FIB): the forwarding table of a node

3. Link-State IGP Segments

Within an SR domain, an SR-capable IGP node advertises segments for its attached prefixes and adjacencies. These segments are called "IGP segments" or "IGP SIDs". They play a key role in Segment Routing and use cases as they enable the expression of any path throughout the SR domain. Such a path is either expressed as a single IGP segment or a list of multiple IGP segments.

Advertisement of IGP segments requires extensions in link-state IGP protocols. These extensions are defined in [ISIS-SR-EXT], [OSPF-SR-EXT], and [OSPFv3-SR-EXT].

3.1. IGP-Prefix Segment (Prefix-SID)

An IGP-Prefix segment is an IGP segment attached to an IGP prefix. An IGP-Prefix segment is global (unless explicitly advertised otherwise) within the SR domain. The context for an IGP-Prefix segment includes the prefix, topology, and algorithm. Multiple SIDs MAY be allocated to the same prefix so long as the tuple <prefix, topology, algorithm> is unique.

Multiple instances and topologies are defined in IS-IS and OSPF in: [RFC5120], [RFC8202], [RFC6549], and [RFC4915].

3.1.1. Prefix-SID Algorithm

Segment Routing supports the use of multiple routing algorithms, i.e, different constraint-based shortest-path calculations can be supported. An algorithm identifier is included as part of a Prefix-SID advertisement. Specification of how an algorithm-specific path calculation is done is required in the document defining the algorithm.

This document defines two algorithms:

- o Shortest Path First: this algorithm is the default behavior. The packet is forwarded along the well known ECMP-aware Shortest Path First (SPF) algorithm employed by the IGPs. However, it is explicitly allowed for a midpoint to implement another forwarding based on local policy. The Shortest Path First algorithm is, in fact, the default and current behavior of most of the networks where local policies may override the SPF decision.

- o Strict Shortest Path First (Strict-SPF): This algorithm mandates that the packet be forwarded according to the ECMP-aware SPF algorithm and instructs any router in the path to ignore any possible local policy overriding the SPF decision. The SID advertised with the Strict-SPF algorithm ensures that the path the packet is going to take is the expected, and not altered, SPF path. Note that Fast Reroute (FRR) [RFC5714] mechanisms are still compliant with the Strict Shortest Path First algorithm. In other words, a packet received with a Strict-SPF SID may be rerouted through an FRR mechanism. Strict-SPF uses the same topology used by the Shortest Path First algorithm. Obviously, nodes that do not support Strict-SPF will not install forwarding entries for this algorithm. Restricting the topology only to those nodes that support this algorithm will not produce the desired forwarding paths since the desired behavior is to follow the path calculated by the Shortest Path First algorithm. Therefore, a source SR node MUST NOT use an SR Policy containing a strict SPF segment if the path crosses a node not supporting the Strict-SPF algorithm.

An IGP-Prefix segment identifies the path, to the related prefix, computed as per the associated algorithm. A packet injected anywhere within the SR domain with an active Prefix-SID is expected to be forwarded along a path computed using the specified algorithm. For this to be possible, a fully connected topology of routers supporting the specified algorithm is required.

3.1.2. SR-MPLS

When SR is used over the MPLS data plane, SIDs are an MPLS label or an index into an MPLS label space (either SRGB or SRLB).

Where possible, it is recommended that identical SRGBs be configured on all nodes in an SR domain. This simplifies troubleshooting as the same label will be associated with the same prefix on all nodes. In addition, it simplifies support for anycast as detailed in Section 3.3.

The following behaviors are associated with SR operating over the MPLS data plane:

- The IGP signaling extension for IGP-Prefix segment includes a flag to indicate whether directly connected neighbors of the node on which the prefix is attached should perform the NEXT operation or the CONTINUE operation when processing the SID. This behavior is equivalent to Penultimate Hop Popping (NEXT) or Ultimate Hop Popping (CONTINUE) in MPLS.

- A Prefix-SID is allocated in the form of an MPLS label (or an index in the SRGB) according to a process similar to IP address allocation. Typically, the Prefix-SID is allocated by policy by the operator (or Network Management System (NMS)), and the SID very rarely changes.

o While SR allows a local segment to be attached to an IGP prefix, where the terminology "IGP-Prefix segment" or "Prefix-SID" is used, the segment is assumed to be global (i.e., the SID is defined from the advertised SRGB). This is consistent with all the described use cases that require global segments attached to IGP prefixes.

o The allocation process MUST NOT allocate the same Prefix-SID to different prefixes.

o If a node learns of a Prefix-SID that has a value that falls outside the locally configured SRGB range, then the node MUST NOT use the Prefix-SID and SHOULD issue an error log reporting a misconfiguration.

o If a node N advertises Prefix-SID SID-R for a prefix R that is attached to N and specifies CONTINUE as the operation to be performed by directly connected neighbors, then N MUST maintain the following FIB entry:

```
   Incoming Active Segment: SID-R
   Ingress Operation: NEXT
   Egress interface: NULL
```

o A remote node M MUST maintain the following FIB entry for any learned Prefix-SID SID-R attached to prefix R:

```
  Incoming Active Segment: SID-R
  Ingress Operation:
     If the next-hop of R is the originator of R
     and M has been instructed to remove the active segment: NEXT
     Else: CONTINUE
  Egress interface: the interface(s) towards the next-hop along the
                    path computed using the algorithm advertised with
                    the SID toward prefix R.
```

As Prefix-SIDs are specific to a given algorithm, if traffic associated with an algorithm arrives at a node that does not support that algorithm, the traffic will be dropped as there will be no forwarding entry matching the incoming label.

3.1.3. SRv6

When SR is used over the IPv6 data plane:

- o A Prefix-SID is an IPv6 address.
- o An operator MUST explicitly instantiate an SRv6 SID. IPv6 node addresses are not SRv6 SIDs by default.

A node N advertising an IPv6 address R usable as a segment identifier MUST maintain the following FIB entry:

```
Incoming Active Segment: R
Ingress Operation: NEXT
Egress interface: NULL
```

Note that forwarding to R does not require an entry in the FIBs of all other routers for R. Forwarding can be, and most often will be, achieved by a shorter mask prefix that covers R.

Independent of SR support, any remote IPv6 node will maintain a plain IPv6 FIB entry for any prefix, no matter if the prefix represents a segment or not. This allows forwarding of packets to the node that owns the SID even by nodes that do not support SR.

Support of multiple algorithms applies to SRv6. Since algorithm-specific SIDs are simply IPv6 addresses, algorithm-specific forwarding entries can be achieved by assigning algorithm-specific subnets to the (set of) algorithm specific SIDs that a node allocates.

Nodes that do not support a given algorithm may still have a FIB entry covering an algorithm-specific address even though an algorithm-specific path has not been calculated by that node. This is mitigated by the fact that nodes that do not support a given algorithm will not be included in the topology associated with that algorithm-specific SPF; therefore, traffic using the algorithm-specific destination will normally not flow via the excluded node. If such traffic were to arrive and be forwarded by such a node, it will still progress towards the destination node. The next-hop will be either a node that supports the algorithm -- in which case, the packet will be forwarded along algorithm-specific paths (or be dropped if none are available) -- or a node that does NOT support the algorithm -- in which case, the packet will continue to be forwarded along Algorithm 0 paths towards the destination node.

3.2. IGP-Node Segment (Node-SID)

An IGP Node-SID MUST NOT be associated with a prefix that is owned by more than one router within the same routing domain.

3.3. IGP-Anycast Segment (Anycast-SID)

An Anycast segment or Anycast-SID enforces the ECMP-aware shortest-path forwarding towards the closest node of the anycast set. This is useful to express macro-engineering policies or protection mechanisms.

An IGP-Anycast segment MUST NOT reference a particular node.

Within an anycast group, all routers in an SR domain MUST advertise the same prefix with the same SID value.

3.3.1. Anycast-SID in SR-MPLS

```
                               +--------------+
                               |   Group A    |
                               |192.0.2.10/32 |
                               |    SID:100   |
                               |              |
                        +-----------A1---A3----------+
                        |      |    | \ / |   |      |
             SID:10     |      |    |  /  |   |      |     SID:30
       203.0.113.1/32   |      |    | / \ |   |      |  203.0.113.3/32
               PE1------R1----------A2---A4---------R3------PE3
                 \     /|      |              |      |\     /
                  \   / |      +--------------+      | \   /
                   \ /  |                            |  \ /
                    /   |                            |   /
                   / \  |                            |  / \
                  /   \ |      +--------------+      | /   \
                 /     \|      |              |      |/     \
               PE2------R2----------B1---B3---------R4------PE4
       203.0.113.2/32   |      |    | \ / |   |      | 203.0.113.4/32
             SID:20     |      |    |  /  |   |      |     SID:40
                        |      |    | / \ |   |      |
                        +-----------B2---B4----------+
                               |              |
                               |   Group B    |
                               | 192.0.2.1/32 |
                               |    SID:200   |
                               +--------------+
```

Figure 1: Transit Device Groups

The Figure 1 illustrates a network example with two groups of transit devices. Group A consists of devices {A1, A2, A3, and A4}. They are all provisioned with the anycast address 192.0.2.10/32 and the Anycast-SID 100.

Similarly, Group B consists of devices {B1, B2, B3, and B4}, and they are all provisioned with the anycast address 192.0.2.1/32 and the Anycast-SID 200. In the above network topology, each Provide Edge (PE) device has a path to each of the groups: A and B.

PE1 can choose a particular transit device group when sending traffic to PE3 or PE4. This will be done by pushing the Anycast-SID of the group in the stack.

Processing the anycast, and subsequent segments, requires special care.

```
                         +--------------------------+
                         |       Group A            |
                         |     192.0.2.10/32        |
                         |        SID:100           |
                         |--------------------------|
                         |                          |
                         |   SRGB:         SRGB:    |
      SID:10             |(1000-2000)    (3000-4000)|             SID:30
        PE1---+       +-------A1-------------A3-------+        +---PE3
               \     /   |    | \           / |     |   \     /
                \   /    |    |  +-----+   /  |     |    \   /
         SRGB:   \ /     |    |         \ /   |     |     \ /   SRGB:
      (7000-8000) R1     |    |          \    |     |      R3 (6000-7000)
                 / \     |    |         / \   |     |     / \
                /   \    |    |  +-----+   \  |     |    /   \
               /     \   |    | /           \ |     |   /     \
        PE2---+       +-------A2-------------A4-------+        +---PE4
      SID:20             |   SRGB:         SRGB:    |             SID:40
                         |(2000-3000)    (4000-5000)|
                         |                          |
                         +--------------------------+
```

Figure 2: Transit Paths via Anycast Group A

Considering an MPLS deployment, in the above topology, if device PE1 (or PE2) requires the sending of a packet to the device PE3 (or PE4), it needs to encapsulate the packet in an MPLS payload with the following stack of labels.

o Label allocated by R1 for Anycast-SID 100 (outer label).

o Label allocated by the nearest router in Group A for SID 30 (for destination PE3).

In this case, the first label is easy to compute. However, because there is more than one device that is topologically nearest (A1 and A2), determining the second label is impossible unless A1 and A2 allocated the same label value to the same prefix. Devices A1 and A2 may be devices from different hardware vendors. If both don’t allocate the same label value for SID 30, it is impossible to use the anycast Group A as a transit anycast group towards PE3. Hence, PE1 (or PE2) cannot compute an appropriate label stack to steer the packet exclusively through the Group A devices. Same holds true for devices PE3 and PE4 when trying to send a packet to PE1 or PE2.

To ease the use of an anycast segment, it is recommended to configure identical SRGBs on all nodes of a particular anycast group. Using this method, as mentioned above, computation of the label following the anycast segment is straightforward.

Using an anycast segment without configuring identical SRGBs on all nodes belonging to the same anycast group may lead to misrouting (in an MPLS VPN deployment, some traffic may leak between VPNs).

3.4. IGP-Adjacency Segment (Adj-SID)

The adjacency is formed by the local node (i.e., the node advertising the adjacency in the IGP) and the remote node (i.e., the other end of the adjacency). The local node MUST be an IGP node. The remote node may be an adjacent IGP neighbor or a non-adjacent neighbor (e.g., a forwarding adjacency, [RFC4206]).

A packet injected anywhere within the SR domain with a segment list {SN, SNL} where SN is the Node-SID of node N and SNL is an Adj-SID attached by node N to its adjacency over link L will be forwarded along the shortest path to N and then be switched by N, without any IP shortest-path consideration, towards link L. If the Adj-SID identifies a set of adjacencies, then the node N load-balances the traffic among the various members of the set.

Similarly, when using a global Adj-SID, a packet injected anywhere within the SR domain with a segment list {SNL}, where SNL is a global Adj-SID attached by node N to its adjacency over link L, will be forwarded along the shortest path to N and then be switched by N, without any IP shortest-path consideration, towards link L. If the Adj-SID identifies a set of adjacencies, then the node N does load-balance the traffic among the various members of the set. The use of global Adj-SID allows to reduce the size of the segment list when expressing a path at the cost of additional state (i.e., the global Adj-SID will be inserted by all routers within the area in their forwarding table).

An "IGP-Adjacency segment" or "Adj-SID" enforces the switching of the packet from a node towards a defined interface or set of interfaces. This is key to theoretically prove that any path can be expressed as a list of segments.

The encodings of the Adj-SID include a set of flags supporting the following functionalities:

o Eligible for Protection (e.g., using IPFRR or MPLS-FRR). Protection allows that in the event the interface(s) associated with the Adj-SID are down, that the packet can still be forwarded via an alternate path. The use of protection is clearly a policy-based decision; that is, for a given policy protection may or may not be desirable.

o Indication whether the Adj-SID has local or global scope. Default scope SHOULD be local.

o Indication whether the Adj-SID is persistent across control plane restarts. Persistence is a key attribute in ensuring that an SR Policy does not temporarily result in misforwarding due to reassignment of an Adj-SID.

A weight (as described below) is also associated with the Adj-SID advertisement.

A node SHOULD allocate one Adj-SID for each of its adjacencies.

A node MAY allocate multiple Adj-SIDs for the same adjacency. An example is to support an Adj-SID that is eligible for protection and an Adj-SID that is NOT eligible for protection.

A node MAY associate the same Adj-SID to multiple adjacencies.

In order to be able to advertise in the IGP all the Adj-SIDs representing the IGP adjacencies between two nodes, parallel adjacency suppression MUST NOT be performed by the IGP.

When a node binds an Adj-SID V to a local data-link L, the node MUST install the following FIB entry:

```
   Incoming Active Segment: V
   Ingress Operation: NEXT
   Egress Interface: L
```

The Adj-SID implies, from the router advertising it, the forwarding of the packet through the adjacency or adjacencies identified by the Adj-SID, regardless of its IGP/SPF cost. In other words, the use of adjacency segments overrides the routing decision made by the SPF algorithm.

3.4.1. Parallel Adjacencies

Adj-SIDs can be used in order to represent a set of parallel interfaces between two adjacent routers.

A node MUST install a FIB entry for any locally originated Adj-SID of value W attached to a set of links B with:

```
   Incoming Active Segment: W
   Ingress Operation: NEXT
   Egress interfaces: load-balance between any data-link within set B
```

When parallel adjacencies are used and associated with the same Adj-SID, and, in order to optimize the load-balancing function, a "weight" factor can be associated with the Adj-SID advertised with each adjacency. The weight tells the ingress (or an SDN/orchestration system) about the load-balancing factor over the parallel adjacencies. As shown in Figure 3, A and B are connected through two parallel adjacencies

```
                    Link-1
                  +--------+
                  |        |
              S---A        B---C
                  |        |
                  +--------+
                    Link-2
```

Figure 3: Parallel Links and Adj-SIDs

Node A advertises following Adj-SIDs and weights:

- o Link-1: Adj-SID 1000, weight: 1
- o Link-2: Adj-SID 1000, weight: 2

Node S receives the advertisements of the parallel adjacencies and understands that by using Adj-SID 1000 node A will load-balance the traffic across the parallel links (Link-1 and Link-2) according to a 1:2 ratio i.e., twice as many packets will flow over Link-2 as compared to Link-1.

3.4.2. LAN Adjacency Segments

In LAN subnetworks, link-state protocols define the concept of Designated Router (DR, in OSPF) or Designated Intermediate System (DIS, in IS-IS) that conduct flooding in broadcast subnetworks and that describe the LAN topology in a special routing update (OSPF Type2 LSA or IS-IS Pseudonode LSP).

The difficulty with LANs is that each router only advertises its connectivity to the DR/DIS and not to each of the individual nodes in the LAN. Therefore, additional protocol mechanisms (IS-IS and OSPF) are necessary in order for each router in the LAN to advertise an Adj-SID associated with each neighbor in the LAN.

3.5. Inter-Area Considerations

In the following example diagram, it is assumed that the all areas are part of a single SR domain.

The Figure 4 assumes the IPv6 control plane with the MPLS data plane.

```
               !          !
               !          !
        B------C-----F----G-----K
       /       |          |     |
 S---A/        |          |     |
      \        |          |     |
       \D------I----------J-----L----Z (2001:DB8::2:1/128, Node-SID 150)
               !          !
       Area 1  ! Backbone ! Area 2
               !   area   !
```

Figure 4: Inter-Area Topology Example

In Area 2, node Z allocates Node-SID 150 to his local IPv6 prefix 2001:DB8::2:1/128.

Area Border Routers (ABRs) G and J will propagate the prefix and its SIDs into the backbone area by creating a new instance of the prefix according to normal inter-area/level IGP propagation rules.

Nodes C and I will apply the same behavior when leaking prefixes from the backbone area down to area 1. Therefore, node S will see prefix 2001:DB8::2:1/128 with Prefix-SID 150 and advertised by nodes C and I.

Therefore, the result is that a Prefix-SID remains attached to its related IGP prefix through the inter-area process, which is the expected behavior in a single SR domain.

When node S sends traffic to 2001:DB8::2:1/128, it pushes Node-SID(150) as an active segment and forwards it to A.

When a packet arrives at ABR I (or C), the ABR forwards the packet according to the active segment (Node-SID(150)). Forwarding continues across area borders, using the same Node-SID(150) until the packet reaches its destination.

4. BGP Segments

BGP segments may be allocated and distributed by BGP.

4.1. BGP-Prefix Segment

A BGP-Prefix segment is a BGP segment attached to a BGP prefix.

A BGP-Prefix segment is global (unless explicitly advertised otherwise) within the SR domain.

The BGP-Prefix segment is the BGP equivalent to the IGP-Prefix segment.

A likely use case for the BGP-Prefix segment is an IGP-free hyper-scale spine-leaf topology where connectivity is learned solely via BGP [RFC7938]

4.2. BGP Peering Segments

In the context of BGP Egress Peer Engineering (EPE), as described in [SR-CENTRAL-EPE], an EPE-enabled egress node MAY advertise segments corresponding to its attached peers. These segments are called BGP peering segments or BGP peering SIDs. They enable the expression of source-routed inter-domain paths.

An ingress border router of an Autonomous System (AS) may compose a list of segments to steer a flow along a selected path within the AS towards a selected egress border router C of the AS and through a specific peer. At a minimum, a BGP peering engineering policy applied at an ingress node involves two segments: the Node-SID of the chosen egress node and the BGP peering segment for the chosen egress node peer or peering interface.

Three types of BGP peering segments/SIDs are defined: PeerNode SID, PeerAdj SID, and PeerSet SID.

- o PeerNode SID: a BGP PeerNode segment/SID is a local segment. At the BGP node advertising it, its semantics are:
  - * SR operation: NEXT.
  - * Next-Hop: the connected peering node to which the segment is related.
- o PeerAdj SID: a BGP PeerAdj segment/SID is a local segment. At the BGP node advertising it, the semantics are:
  - * SR operation: NEXT.
  - * Next-Hop: the peer connected through the interface to which the segment is related.
- o PeerSet SID: a BGP PeerSet segment/SID is a local segment. At the BGP node advertising it, the semantics are:
  - * SR operation: NEXT.
  - * Next-Hop: load-balance across any connected interface to any peer in the related group.

  A peer set could be all the connected peers from the same AS or a subset of these. A group could also span across AS. The group definition is a policy set by the operator.

The BGP extensions necessary in order to signal these BGP peering segments are defined in [BGPLS-SR-EPE].

5. Binding Segment

In order to provide greater scalability, network opacity, and service independence, SR utilizes a Binding SID (BSID). The BSID is bound to an SR Policy, instantiation of which may involve a list of SIDs. Any packets received with an active segment equal to BSID are steered onto the bound SR Policy.

A BSID may be either a local or a global SID. If local, a BSID SHOULD be allocated from the SRLB. If global, a BSID MUST be allocated from the SRGB.

Use of a BSID allows the instantiation of the policy (the SID list) to be stored only on the node or nodes that need to impose the policy. Direction of traffic to a node supporting the policy then only requires imposition of the BSID. If the policy changes, this also means that only the nodes imposing the policy need to be updated. Users of the policy are not impacted.

5.1. IGP Mirroring Context Segment

One use case for a Binding segment is to provide support for an IGP node to advertise its ability to process traffic originally destined to another IGP node, called the "mirrored node" and identified by an IP address or a Node-SID, provided that a Mirroring Context segment is inserted in the segment list prior to any service segment local to the mirrored node.

When a given node B wants to provide egress node A protection, it advertises a segment identifying node’s A context. Such a segment is called "Mirroring Context segment" and is identified by the Mirror SID.

The Mirror SID is advertised using the Binding segment defined in SR IGP protocol extensions [ISIS-SR-EXT].

In the event of a failure, a Point of Local Repair (PLR) diverting traffic from A to B does a PUSH of the Mirror SID on the protected traffic. When receiving the traffic with the Mirror SID as the active segment, B uses that segment and processes underlying segments in the context of A.

6. Multicast

Segment Routing is defined for unicast. The application of the source-route concept to Multicast is not in the scope of this document.

7. IANA Considerations

This document has no IANA actions.

8. Security Considerations

Segment Routing is applicable to both MPLS and IPv6 data planes.

SR adds some metadata (instructions) to the packet, with the list of forwarding path elements (e.g., nodes, links, services, etc.) that the packet must traverse. It has to be noted that the complete source-routed path may be represented by a single segment. This is the case of the Binding SID.

By default, SR operates within a trusted domain. Traffic MUST be filtered at the domain boundaries.

The use of best practices to reduce the risk of tampering within the trusted domain is important. Such practices are discussed in [RFC4381] and are applicable to both SR-MPLS and SRv6.

8.1. SR-MPLS

When applied to the MPLS data plane, SR does not introduce any new behavior or any change in the way the MPLS data plane works. Therefore, from a security standpoint, this document does not define any additional mechanism in the MPLS data plane.

SR allows the expression of a source-routed path using a single segment (the Binding SID). Compared to RSVP-TE, which also provides explicit routing capability, there are no fundamental differences in terms of information provided. Both RSVP-TE and Segment Routing may express a source-routed path using a single segment.

When a path is expressed using a single label, the syntax of the metadata is equivalent between RSVP-TE [RFC3209] and SR.

When a source-routed path is expressed with a list of segments, additional metadata is added to the packet consisting of the source-routed path the packet must follow expressed as a segment list.

When a path is expressed using a label stack, if one has access to the meaning (i.e., the Forwarding Equivalence Class) of the labels, one has the knowledge of the explicit path. For the MPLS data plane, as no data-plane modification is required, there is no fundamental change of capability. Yet, the occurrence of label stacking will increase.

SR domain boundary routers MUST filter any external traffic destined to a label associated with a segment within the trusted domain. This includes labels within the SRGB of the trusted domain, labels within the SRLB of the specific boundary router, and labels outside either of these blocks. External traffic is any traffic received from an interface connected to a node outside the domain of trust.

From a network protection standpoint, there is an assumed trust model such that any node imposing a label stack on a packet is assumed to be allowed to do so. This is a significant change compared to plain IP offering shortest path routing, but it is not fundamentally different compared to existing techniques providing explicit routing capability such as RSVP-TE. By default, the explicit routing information MUST NOT be leaked through the boundaries of the administered domain. Segment Routing extensions that have been defined in various protocols, leverage the security mechanisms of these protocols such as encryption, authentication, filtering, etc.

In the general case, a segment-routing-capable router accepts and installs labels only if the labels have been previously advertised by a trusted source. The received information is validated using existing control-plane protocols providing authentication and security mechanisms. Segment Routing does not define any additional security mechanism in existing control-plane protocols.

SR does not introduce signaling between the source and the midpoints of a source-routed path. With SR, the source-routed path is computed using SIDs previously advertised in the IP control plane. Therefore, in addition to filtering and controlled advertisement of SIDs at the boundaries of the SR domain, filtering in the data plane is also required. Filtering MUST be performed on the forwarding plane at the boundaries of the SR domain and may require looking at multiple labels/instructions.

For the MPLS data plane, there are no new requirements as the existing MPLS architecture already allows such source routing by stacking multiple labels. And, for security protection, [RFC4381] and [RFC5920] already call for the filtering of MPLS packets on trust boundaries.

8.2. SRv6

When applied to the IPv6 data plane, Segment Routing does introduce the Segment Routing Header (SRH, [IPv6-SRH]) which is a type of Routing Extension header as defined in [RFC8200].

The SRH adds some metadata to the IPv6 packet, with the list of forwarding path elements (e.g., nodes, links, services, etc.) that the packet must traverse and that are represented by IPv6 addresses. A complete source-routed path may be encoded in the packet using a single segment (single IPv6 address).

SR domain boundary routers MUST filter any external traffic destined to an address within the SRGB of the trusted domain or the SRLB of the specific boundary router. External traffic is any traffic received from an interface connected to a node outside the domain of trust.

From a network-protection standpoint, there is an assumed trust model such that any node adding an SRH to the packet is assumed to be allowed to do so. Therefore, by default, the explicit routing information MUST NOT be leaked through the boundaries of the administered domain. Segment Routing extensions that have been defined in various protocols, leverage the security mechanisms of these protocols such as encryption, authentication, filtering, etc.

In the general case, an SRv6 router accepts and install segments identifiers (in the form of IPv6 addresses), only if these SIDs are advertised by a trusted source. The received information is validated using existing control-plane protocols providing authentication and security mechanisms. Segment Routing does not define any additional security mechanism in existing control-plane protocols.

Problems that may arise when the above behaviors are not implemented or when the assumed trust model is violated (e.g., through a security breach) include:

- Malicious looping

- Evasion of access controls

- Hiding the source of DoS attacks

Security concerns with SR at the IPv6 data plane are more completely discussed in [RFC5095]. The new IPv6-based Segment Routing Header is defined in [IPv6-SRH]. This document also discusses the above security concerns.

8.3. Congestion Control

SR does not introduce new requirements for congestion control. By default, traffic delivery is assumed to be best effort. Congestion control may be implemented at endpoints. Where SR policies are in use, bandwidth allocation may be managed by monitoring incoming traffic associated with the binding SID identifying the SR Policy. Other solutions such as presented in [RFC8084] may be applicable.

9. Manageability Considerations

In SR-enabled networks, the path the packet takes is encoded in the header. As the path is not signaled through a protocol, OAM mechanisms are necessary in order for the network operator to validate the effectiveness of a path as well as to check and monitor its liveness and performance. However, it has to be noted that SR allows to reduce substantially the number of states in transit nodes; hence, the number of elements that a transit node has to manage is smaller.

SR OAM use cases for the MPLS data plane are defined in [RFC8403]. SR OAM procedures for the MPLS data plane are defined in [RFC8287].

SR routers receive advertisements of SIDs (index, label, or IPv6 address) from the different routing protocols being extended for SR. Each of these protocols have monitoring and troubleshooting mechanisms to provide operation and management functions for IP addresses that must be extended in order to include troubleshooting and monitoring functions of the SID.

SR architecture introduces the usage of global segments. Each global segment MUST be bound to a unique index or address within an SR domain. The management of the allocation of such an index or address by the operator is critical for the network behavior to avoid situations like misrouting. In addition to the allocation policy/ tooling that the operator will have in place, an implementation SHOULD protect the network in case of conflict detection by providing a deterministic resolution approach.

When a path is expressed using a label stack, the occurrence of label stacking will increase. A node may want to signal, in the control plane, its ability in terms of size of the label stack it can support.

A YANG data model [RFC6020] for SR configuration and operations has been defined in [SR-YANG].

When SR is applied to the IPv6 data plane, segments are identified through IPv6 addresses. The allocation, management, and troubleshooting of segment identifiers is no different than the existing mechanisms applied to the allocation and management of IPv6 addresses.

The DA of the packet gives the active segment address. The segment list in the SRH gives the entire path of the packet. The validation of the source-routed path is done through inspection of DA and SRH present in the packet header matched to the equivalent routing table entries.

In the context of the SRv6 data plane, the source-routed path is encoded in the SRH as described in [IPv6-SRH]. The SRv6 source-routed path is instantiated into the SRH as a list of IPv6 addresses where the active segment is in the DA field of the IPv6 packet header. Typically, by inspecting, in any node, the packet header, it is possible to derive the source-routed path to which it belongs. Similar to the context of the SR-MPLS data plane, an implementation may originate path control and monitoring packets where the source-routed path is inserted in the SRH and where each segment of the path inserts in the packet the relevant data in order to measure the end-to-end path and performance.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, DOI 10.17487/RFC3031, January 2001, <https://www.rfc-editor.org/info/rfc3031>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

10.2. Informative References

[BGPLS-SR-EPE]
Previdi, S., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", Work in Progress, draft-ietf-idr-bgpls-segment-routing-epe-15, March 2018.

[IPv6-SRH]
Filsfils, C., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, Ed., "IPv6 Segment Routing Header (SRH)", Work in Progress, draft-ietf-6man-segment-routing-header-14, June 2018.

[ISIS-SR-EXT]
Previdi, S., Ed., Ginsberg, L., Ed., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", Work in Progress, draft-ietf-isis-segment-routing-extensions-19, July 2018.

[OSPF-SR-EXT]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", Work in Progress, draft-ietf-ospf-segment-routing-extensions-25, April 2018.

[OSPFv3-SR-EXT]
Psenak, P., Ed., Filsfils, C., Previdi, S., Ed., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPFv3 Extensions for Segment Routing", Work in Progress, draft-ietf-ospf-ospfv3-segment-routing-extensions-13, May 2018.

[PCEP-SR-EXT]
Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "PCEP Extensions for Segment Routing", Work in Progress, draft-ietf-pce-segment-routing-12, June 2018.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

[RFC4206] Kompella, K. and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, DOI 10.17487/RFC4206, October 2005, <https://www.rfc-editor.org/info/rfc4206>.

[RFC4381] Behringer, M., "Analysis of the Security of BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4381, DOI 10.17487/RFC4381, February 2006, <https://www.rfc-editor.org/info/rfc4381>.

[RFC4915] Psenak, P., Mirtorabi, S., Roy, A., Nguyen, L., and P. Pillay-Esnault, "Multi-Topology (MT) Routing in OSPF", RFC 4915, DOI 10.17487/RFC4915, June 2007, <https://www.rfc-editor.org/info/rfc4915>.

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <https://www.rfc-editor.org/info/rfc5095>.

[RFC5120] Przygienda, T., Shen, N., and N. Sheth, "M-ISIS: Multi Topology (MT) Routing in Intermediate System to Intermediate Systems (IS-ISs)", RFC 5120, DOI 10.17487/RFC5120, February 2008, <https://www.rfc-editor.org/info/rfc5120>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440, DOI 10.17487/RFC5440, March 2009, <https://www.rfc-editor.org/info/rfc5440>.

[RFC5714] Shand, M. and S. Bryant, "IP Fast Reroute Framework", RFC 5714, DOI 10.17487/RFC5714, January 2010, <https://www.rfc-editor.org/info/rfc5714>.

[RFC5920] Fang, L., Ed., "Security Framework for MPLS and GMPLS Networks", RFC 5920, DOI 10.17487/RFC5920, July 2010, <https://www.rfc-editor.org/info/rfc5920>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC6549] Lindem, A., Roy, A., and S. Mirtorabi, "OSPFv2 Multi-Instance Extensions", RFC 6549, DOI 10.17487/RFC6549, March 2012, <https://www.rfc-editor.org/info/rfc6549>.

[RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <https://www.rfc-editor.org/info/rfc7938>.

[RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <https://www.rfc-editor.org/info/rfc8084>.

[RFC8202] Ginsberg, L., Previdi, S., and W. Henderickx, "IS-IS Multi-Instance", RFC 8202, DOI 10.17487/RFC8202, June 2017, <https://www.rfc-editor.org/info/rfc8202>.

[RFC8287] Kumar, N., Ed., Pignataro, C., Ed., Swallow, G., Akiya, N., Kini, S., and M. Chen, "Label Switched Path (LSP) Ping/Traceroute for Segment Routing (SR) IGP-Prefix and IGP-Adjacency Segment Identifiers (SIDs) with MPLS Data Planes", RFC 8287, DOI 10.17487/RFC8287, December 2017, <https://www.rfc-editor.org/info/rfc8287>.

[RFC8355] Filsfils, C., Ed., Previdi, S., Ed., Decraene, B., and R. Shakir, "Resiliency Use Cases in Source Packet Routing in Networking (SPRING) Networks", RFC 8355, DOI 10.17487/RFC8355, March 2018, <https://www.rfc-editor.org/info/rfc8355>.

[RFC8403] Geib, R., Ed., Filsfils, C., Pignataro, C., Ed., and N. Kumar, "A Scalable and Topology-Aware MPLS Data-Plane Monitoring System", RFC 8403, DOI 10.17487/RFC8403, July 2018, <http://www.rfc-editor.org/info/rfc8403>.

[SR-CENTRAL-EPE]
Filsfils, C., Previdi, S., Dawra, G., Aries, E., and D. Afanasiev, "Segment Routing Centralized BGP Egress Peer Engineering", Work in Progress, draft-ietf-spring-segment-routing-central-epe-10, December 2017.

[SR-MPLS] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with MPLS data plane", Work in Progress, draft-ietf-spring-segment-routing-mpls-14, June 2018.

[SR-YANG] Litkowski, S., Qu, Y., Sarkar, P., and J. Tantsura, "YANG Data Model for Segment Routing", Work in Progress, draft-ietf-spring-sr-yang-09, June 2018.

Acknowledgements

We would like to thank Dave Ward, Peter Psenak, Dan Frost, Stewart Bryant, Pierre Francois, Thomas Telkamp, Ruediger Geib, Hannes Gredler, Pushpasis Sarkar, Eric Rosen, Chris Bowers, and Alvaro Retana for their comments and review of this document.

Contributors

The following people have substantially contributed to the definition of the Segment Routing architecture and to the editing of this document:

Ahmed Bashandy
Cisco Systems, Inc.
Email: bashandy@cisco.com

Martin Horneffer
Deutsche Telekom
Email: Martin.Horneffer@telekom.de

Wim Henderickx
Nokia
Email: wim.henderickx@nokia.com

Jeff Tantsura
Email: jefftant@gmail.com

Edward Crabbe
Email: edward.crabbe@gmail.com

Igor Milojevic
Email: milojevicigor@gmail.com

Saku Ytti
TDC
Email: saku@ytti.fi

Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
Belgium

Email: cfilsfil@cisco.com

Stefano Previdi (editor)
Cisco Systems, Inc.
Italy

Email: stefano@previdi.net

Les Ginsberg
Cisco Systems, Inc.

Email: ginsberg@cisco.com

Bruno Decraene
Orange
FR

Email: bruno.decraene@orange.com

Stephane Litkowski
Orange
France

Email: stephane.litkowski@orange.com

Rob Shakir
Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94043
United States of America

Email: robjs@google.com

ANNEX 2 – RFC PROPOSED STANDARD ON "IPV6 SEGMENT ROUTING HEADER"

```
Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 24, 2020
```

```
                                                    C. Filsfils, Ed.
                                                      D. Dukes, Ed.
                                                Cisco Systems, Inc.
                                                         S. Previdi
                                                             Huawei
                                                           J. Leddy
                                                         Individual
                                                      S. Matsushima
                                                           Softbank
                                                           D. Voyer
                                                        Bell Canada
                                                   October 22, 2019
```

IPv6 Segment Routing Header (SRH)
draft-ietf-6man-segment-routing-header-26

Abstract

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Extension Header called the Segment Routing Header. This document describes the Segment Routing Header and how it is used by Segment Routing capable nodes.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Segment Routing can be applied to the IPv6 data plane using a new type of Routing Header called the Segment Routing Header. This document describes the Segment Routing Header and how it is used by Segment Routing capable nodes.

The Segment Routing Architecture [RFC8402] describes Segment Routing and its instantiation in two data planes; MPLS and IPv6.

The encoding of IPv6 segments in the Segment Routing Header is defined in this document.

This document uses the terms Segment Routing, SR Domain, SRv6, Segment ID (SID), SRv6 SID, Active Segment, and SR Policy as defined in [RFC8402].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Segment Routing Header

Routing Headers are defined in [RFC8200]. The Segment Routing Header has a new Routing Type (suggested value 4) to be assigned by IANA.

The Segment Routing Header (SRH) is defined as follows:

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    | Next Header   |  Hdr Ext Len  | Routing Type  | Segments Left |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Last Entry   |     Flags     |              Tag              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |            Segment List[0] (128 bits IPv6 address)            |
    |                                                               |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |                                                               |
                                  ...
    |                                                               |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    |            Segment List[n] (128 bits IPv6 address)            |
    |                                                               |
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    //                                                             //
    //         Optional Type Length Value objects (variable)       //
    //                                                             //
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

o Next Header: Defined in [RFC8200] Section 4.4

o Hdr Ext Len: Defined in [RFC8200] Section 4.4

o Routing Type: TBD, to be assigned by IANA (suggested value: 4).

o Segments Left: Defined in [RFC8200] Section 4.4

o Last Entry: contains the index (zero based), in the Segment List, of the last element of the Segment List.

o Flags: 8 bits of flags. Section 8.1 creates an IANA registry for new flags to be defined. The following flags are defined:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
|U U U U U U U U|
+-+-+-+-+-+-+-+-+
```

U: Unused and for future use. MUST be 0 on transmission and ignored on receipt.

o Tag: tag a packet as part of a class or group of packets, e.g., packets sharing the same set of properties. When tag is not used at source it MUST be set to zero on transmission. When tag is not used during SRH Processing it SHOULD be ignored. Tag is not used when processing the SID defined in Section 4.3.1. It may be used when processing other SIDs that are not defined in this document. The allocation and use of tag is outside the scope of this document.

o Segment List[n]: 128 bit IPv6 addresses representing the nth segment in the Segment List. The Segment List is encoded starting from the last segment of the SR Policy. I.e., the first element of the segment list (Segment List [0]) contains the last segment of the SR Policy, the second element contains the penultimate segment of the SR Policy and so on.

o Type Length Value (TLV) are described in Section 2.1.

In the SRH, the Next Header, Hdr Ext Len, Routing Type, and Segments Left fields are defined in Section 4.4 of [RFC8200]. Based on the constraints in that section, Next Header, Header Ext Len, and Routing Type are not mutable while Segments Left is mutable.

The mutability of the TLV value is defined by the most significant bit in the type, as specified in Section 2.1.

Section 4.3 defines the mutability of the remaining fields in the SRH (Flags, Tag, Segment List) in the context of the SID defined in this document.

New SIDs defined in the future MUST specify the mutability properties of the Flags, Tag, and Segment List and indicate how the HMAC TLV (Section 2.1.2) verification works. Note, that in effect these fields are mutable.

Consistent with the source routing model, the source of the SRH always knows how to set the segment list, Flags, Tag and TLVs of the SRH for use within the SR Domain. How it achieves this is outside the scope of this document, but may be based on topology, available SIDs and their mutability properties, the SRH mutability requirements of the destination, or any other information.

2.1. SRH TLVs

This section defines TLVs of the Segment Routing Header.

A TLV provides meta-data for segment processing. The only TLVs defined in this document are the HMAC (Section 2.1.2) and PAD (Section 2.1.1) TLVs. While processing the SID defined in Section 4.3.1, all TLVs are ignored unless local configuration indicates otherwise (Section 4.3.1.1.1). Thus, TLV and HMAC support is optional for any implementation, however, an implementation adding or parsing TLVs MUST support PAD TLVs. Other documents may define additional TLVs and processing rules for them.

TLVs are present when the Hdr Ext Len is greater than (Last Entry+1)*2.

While processing TLVs at a segment endpoint, TLVs MUST be fully contained within the SRH as determined by the Hdr Ext Len. Detection of TLVs exceeding the boundary of the SRH Hdr Ext Len results in an ICMP Parameter Problem, Code 0, message to the Source Address, pointing to the Hdr Ext Len field of the SRH, and the packet being discarded.

An implementation MAY limit the number and/or length of TLVs it processes based on local configuration. It MAY:

- o Limit the number of consecutive Pad1 (Section 2.1.1.1) options to 1. If padding of more than one byte is required, then PadN (Section 2.1.1.2) should be used.
- o Limit the length in PadN to 5.
- o Limit the maximum number of non-Pad TLVs to be processed.
- o Limit the maximum length of all TLVs to be processed.

The implementation MAY stop processing additional TLVs in the SRH when these configured limits are exceeded.

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-----------------------
|     Type      |    Length     | Variable length data
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-----------------------
```

Type: An 8 bit codepoint from Segment Routing Header TLVs Registry TBD IANA Reference. Unrecognized Types MUST be ignored on receipt.

Length: The length of the Variable length data in bytes.

Variable length data: Length bytes of data that is specific to the Type.

Type Length Value (TLV) entries contain OPTIONAL information that may be used by the node identified in the Destination Address (DA) of the packet.

Each TLV has its own length, format and semantic. The codepoint allocated (by IANA) to each TLV Type defines both the format and the semantic of the information carried in the TLV. Multiple TLVs may be encoded in the same SRH.

The highest-order bit of the TLV type (bit 0) specifies whether or not the TLV data of that type can change en route to the packet’s final destination:

0: TLV data does not change en route

1: TLV data does change en route

All TLVs specify their alignment requirements using an xn+y format. The xn+y format is defined as per [RFC8200]. The SR Source nodes use the xn+y alignment requirements of TLVs and Padding TLVs when constructing an SRH.

The "Length" field of the TLV is used to skip the TLV while inspecting the SRH in case the node doesn’t support or recognize the Type. The "Length" defines the TLV length in octets, not including the "Type" and "Length" fields.

The following TLVs are defined in this document:

Padding TLVs

HMAC TLV

Additional TLVs may be defined in the future.

2.1.1. Padding TLVs

There are two types of Padding TLVs, pad1 and padN, the following applies to both:

Padding TLVs are used for meeting the alignment requirement of the subsequent TLVs.

Padding TLVs are used to pad the SRH to a multiple of 8 octets.

Padding TLVs are ignored by a node processing the SRH TLV.

Multiple Padding TLVs MAY be used in one SRH

2.1.1.1. PAD1

Alignment requirement: none

```
  0 1 2 3 4 5 6 7
 +-+-+-+-+-+-+-+-+
 |     Type      |
 +-+-+-+-+-+-+-+-+
```

Type: to be assigned by IANA (Suggested value 0)

A single Pad1 TLV MUST be used when a single byte of padding is required. A Pad1 TLV MUST NOT be used if more than one consecutive byte of padding is required.

2.1.1.2. PADN

Alignment requirement: none

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |    Length     |      Padding (variable)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
//                    Padding (variable)                        //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type: to be assigned by IANA (suggested value 4).

Length: 0 to 5

Padding: Length octets of padding. Padding bits have no semantic. They MUST be set to 0 on transmission and ignored on receipt.

The PadN TLV MUST be used when more than one byte of padding is required.

2.1.2. HMAC TLV

Alignment requirement: 8n

The keyed Hashed Message Authentication Code (HMAC) TLV is OPTIONAL and has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Type     |     Length    |D|        RESERVED             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      HMAC Key ID (4 octets)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                              //
|                      HMAC (Variable)                         //
|                                                              //
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

where:

o Type: to be assigned by IANA (suggested value 5).

o Length: The length of the variable length data in bytes.

o D: 1 bit. 1 indicates the Destination Address verification is disabled due to use of reduced segment list, Section 4.1.1.

o RESERVED: 15 bits. MUST be 0 on transmission.

o HMAC Key ID: A 4 octet opaque number which uniquely identifies the pre-shared key and algorithm used to generate the HMAC.

o HMAC: Keyed HMAC, in multiples of 8 octets, at most 32 octets.

The HMAC TLV is used to verify that the SRH applied to a packet was selected by an authorized party, and to ensure that the segment list is not modified after generation. This also allows for verification that the current segment (by virtue of being in the authorized segment list) is authorized for use. The SR Domain ensures the source node is permitted to use the source address in the packet via ingress filtering mechanisms as defined in BCP 84 [RFC3704], or other strategies as appropriate.

2.1.2.1. HMAC Generation and Verification

Local configuration determines when to check for an HMAC. This local configuration is outside the scope of this document. It may be based on the active segment at an SR Segment endpoint node, the result of an ACL that considers incoming interface, HMAC Key ID, or other packet fields.

An implementation that supports the generation and verification of the HMAC supports the following default behavior, as defined in the remainder of this section.

The HMAC verification begins by checking the current segment is equal to the destination address of the IPv6 header. The check is successful when either

- o HMAC D bit is 1 and Segments Left is greater than Last Entry.
- o HMAC Segments Left is less than or equal to Last Entry and destination address is equal to Segment List [Segments Left].

The HMAC field is the output of the HMAC computation as defined in [RFC2104], using:

- o key: the pre-shared key identified by HMAC Key ID
- o HMAC algorithm: identified by the HMAC Key ID
- o Text: a concatenation of the following fields from the IPv6 header and the SRH, as it would be received at the node verifying the HMAC:
  - * IPv6 header: source address (16 octets)
  - * SRH: Last Entry (1 octet)
  - * SRH: Flags (1 octet)
  - * SRH: HMAC 16 bits following Length
  - * SRH: HMAC Key ID (4 octets)
  - * SRH: all addresses in the Segment List (variable octets)

The HMAC digest is truncated to 32 octets and placed in the HMAC field of the HMAC TLV.

For HMAC algorithms producing digests less than 32 octets, the digest is placed in the lowest order octets of the HMAC field. Subsequent octets MUST be set to zero such that the HMAC length is a multiple of 8 octets.

If HMAC verification is successful, processing proceeds as normal.

If HMAC verification fails, an ICMP error message (parameter problem, error code 0, pointing to the HMAC TLV) SHOULD be generated (but rate limited) and SHOULD be logged and the packet discarded.

2.1.2.2. HMAC Pre-Shared Key Algorithm

The HMAC Key ID field allows for the simultaneous existence of several hash algorithms (SHA-256, SHA3-256 ... or future ones) as well as pre-shared keys.

The HMAC Key ID field is opaque, i.e., it has neither syntax nor semantic except as an identifier of the right combination of pre-shared key and hash algorithm.

At the HMAC TLV generating and verification nodes, the Key ID uniquely identifies the pre-shared key and HMAC algorithm.

At the HMAC TLV generating node, the Text for the HMAC computation is set to the IPv6 header fields and SRH fields as they would appear at the verification node(s), not necessarily the same as the source node sending a packet with the HMAC TLV.

Pre-shared key roll-over is supported by having two key IDs in use while the HMAC TLV generating node and verifying node converge to a new key.

The HMAC TLV generating node may need to revoke an SRH for which it previously generated an HMAC. Revocation is achieved by allocating a new key and key ID, then rolling over the key ID associated with the SRH to be revoked. The HMAC TLV verifying node drops packets with the revoked SRH.

An implementation supporting HMAC can support multiple hash functions. An implementation supporting HMAC MUST implement SHA-2 [FIPS180-4] in its SHA-256 variant.

The selection of pre-shared key and algorithm, and their distribution is outside the scope of this document. Some options may include:

- o in the configuration of the HMAC generating or verifying nodes, either by static configuration or any SDN oriented approach o dynamically using a trusted key distribution protocol such as [RFC6407]

While key management is outside the scope of this document, the recommendations of BCP 107 [RFC4107] should be considered when choosing the key management system.

3. SR Nodes

There are different types of nodes that may be involved in segment routing networks: source SR nodes originate packets with a segment in the destination address of the IPv6 header, transit nodes that forward packets destined to a remote segment, and SR segment endpoint nodes that process a local segment in the destination address of an IPv6 header.

3.1. Source SR Node

A Source SR Node is any node that originates an IPv6 packet with a segment (i.e. SRv6 SID) in the destination address of the IPv6 header. The packet leaving the source SR Node may or may not contain an SRH. This includes either:

A host originating an IPv6 packet.

An SR domain ingress router encapsulating a received packet in an outer IPv6 header, followed by an optional SRH.

The mechanism through which a segment in the destination address of the IPv6 header and the Segment List in the SRH, is derived is outside the scope of this document.

3.2. Transit Node

A transit node is any node forwarding an IPv6 packet where the destination address of that packet is not locally configured as a segment nor a local interface. A transit node is not required to be capable of processing a segment nor SRH.

3.3. SR Segment Endpoint Node

A SR segment endpoint node is any node receiving an IPv6 packet where the destination address of that packet is locally configured as a segment or local interface.

4. Packet Processing

This section describes SRv6 packet processing at the SR source, Transit and SR segment endpoint nodes.

4.1. Source SR Node

A Source node steers a packet into an SR Policy. If the SR Policy results in a segment list containing a single segment, and there is no need to add information to the SRH flag or to add TLV, the DA is set to the single segment list entry and the SRH MAY be omitted.

When needed, the SRH is created as follows:

Next Header and Hdr Ext Len fields are set as specified in [RFC8200].

Routing Type field is set as TBD (to be allocated by IANA, suggested value 4).

The DA of the packet is set with the value of the first segment.

The first element of the SRH Segment List is the ultimate segment. The second element is the penultimate segment, and so on.

The Segments Left field is set to n-1 where n is the number of elements in the SR Policy.

The Last Entry field is set to n-1 where n is the number of elements in the SR Policy.

TLVs (including HMAC) may be set according to their specification.

The packet is forwarded toward the packet’s Destination Address (the first segment).

4.1.1. Reduced SRH

When a source does not require the entire SID list to be preserved in the SRH, a reduced SRH may be used.

A reduced SRH does not contain the first segment of the related SR Policy (the first segment is the one already in the DA of the IPv6 header), and the Last Entry field is set to n-2 where n is the number of elements in the SR Policy.

4.2. Transit Node

As specified in [RFC8200], the only node allowed to inspect the Routing Extension Header (and therefore the SRH), is the node corresponding to the DA of the packet. Any other transit node MUST NOT inspect the underneath routing header and MUST forward the packet toward the DA according to its IPv6 routing table.

When a SID is in the destination address of an IPv6 header of a packet, it’s routed through an IPv6 network as an IPv6 address. SIDs, or the prefix(es) covering SIDs, and their reachability may be distributed by means outside the scope of this document. For example, [RFC5308] or [RFC5340] may be used to advertise a prefix covering the SIDs on a node.

4.3. SR Segment Endpoint Node

Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.

When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packets destination address. This lookup can return any of the following:

```
    * A FIB entry that represents a locally instantiated SRv6 SID
    * A FIB entry that represents a local interface, not locally
                                    instantiated as an SRv6 SID
    * A FIB entry that represents a non-local route
    * No Match
```

4.3.1. FIB Entry Is Locally Instantiated SRv6 SID

This document, and section, defines a single SRv6 SID. Future documents may define additional SRv6 SIDs. In which case, the entire content of this section will be defined in that document.

If the FIB entry represents a locally instantiated SRv6 SID, process the next header chain of the IPv6 header as defined in section 4 of [RFC8200]. Section 4.3.1.1 describes how to process an SRH, Section 4.3.1.2 describes how to process an upper layer header or no next header.

Processing this SID modifies the Segments Left and, if configured to process TLVs, it may modify the "variable length data" of TLV types that change en route. Therefore Segments Left is mutable and TLVs that change en route are mutable. The remainder of the SRH (Flags, Tag, Segment List, and TLVs that do not change en route) are immutable while processing this SID.

4.3.1.1. SRH Processing

```
S01. When an SRH is processed {
S02.   If Segments Left is equal to zero {
S03.     Proceed to process the next header in the packet,
         whose type is identified by the Next Header field in
         the Routing header.
S04.   }
S05.   Else {
S06.     If local configuration requires TLV processing {
S07.       Perform TLV processing (see TLV Processing)
S08.     }
S09.     max_last_entry  =  ( Hdr Ext Len /  2 ) - 1
S10.     If  ((Last Entry > max_last_entry) or
S11.          (Segments Left is greater than (Last Entry+1)) {
S12.       Send an ICMP Parameter Problem, Code 0, message to
           the Source Address, pointing to the Segments Left
           field, and discard the packet.
S13.     }
S14.     Else {
S15.       Decrement Segments Left by 1.
S16.       Copy Segment List[Segments Left] from the SRH to the
           destination address of the IPv6 header.
S17.       If the IPv6 Hop Limit is less than or equal to 1 {
S18.         Send an ICMP Time Exceeded -- Hop Limit Exceeded in
             Transit message to the Source Address and discard
             the packet.
S19.       }
S20.       Else {
S21.         Decrement the Hop Limit by 1
S22.         Resubmit the packet to the IPv6 module for transmission
             to the new destination.
S23.       }
S24.     }
S25.   }
S26. }
```

4.3.1.1.1. TLV Processing

Local configuration determines how TLVs are to be processed when the Active Segment is a local SID defined in this document. The definition of local configuration is outside the scope of this document.

For illustration purpose only, two example local configurations that may be associated with a SID are provided below.

```
Example 1:
For any packet received from interface I2
  Skip TLV processing

Example 2:
For any packet received from interface I1
  If first TLV is HMAC {
    Process the HMAC TLV
  }
  Else {
    Discard the packet
  }
```

4.3.1.2. Upper-layer Header or No Next Header

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 SID defined in this document.

```
IF (Upper-layer Header is IPv4 or IPv6) and
    local configuration permits {
  Perform IPv6 decapsulation
  Resubmit the decapsulated packet to the IPv4 or IPv6 module
}
ELSE {
  Send an ICMP parameter problem message to the Source Address and
  discard the packet.  Error code (TBD by IANA) "SR Upper-layer
  Header Error", pointer set to the offset of the upper-layer
  header.
}
```

A unique error code allows an SR Source node to recognize an error in SID processing at an endpoint.

4.3.2. FIB Entry Is A Local Interface

If the FIB entry represents a local interface, not locally instantiated as an SRv6 SID, the SRH is processed as follows:

If Segments Left is zero, the node must ignore the Routing header and proceed to process the next header in the packet, whose type is identified by the Next Header field in the Routing Header.

If Segments Left is non-zero, the node must discard the packet and send an ICMP Parameter Problem, Code 0, message to the packet’s Source Address, pointing to the unrecognized Routing Type.

4.3.3. FIB Entry Is A Non-Local Route

Processing is not changed by this document.

4.3.4. FIB Entry Is A No Match

Processing is not changed by this document.

5. Intra SR Domain Deployment Model

The use of the SIDs exclusively within the SR Domain and solely for packets of the SR Domain is an important deployment model.

This enables the SR Domain to act as a single routing system.

This section covers:

- o securing the SR Domain from external attempt to use its SIDs

- o SR Domain as a single system with delegation between components

- o handling packets of the SR Domain

5.1. Securing the SR Domain

Nodes outside the SR Domain are not trusted: they cannot directly use the SIDs of the domain. This is enforced by two levels of access control lists:

1. Any packet entering the SR Domain and destined to a SID within the SR Domain is dropped. This may be realized with the following logic. Other methods with equivalent outcome are considered compliant:

    * allocate all the SID’s from a block S/s

    * configure each external interface of each edge node of the domain with an inbound infrastructure access list (IACL) which drops any incoming packet with a destination address in S/s

    * Failure to implement this method of ingress filtering exposes the SR Domain to source routing attacks as described and referenced in [RFC5095]

2. The distributed protection in #1 is complemented with per node protection, dropping packets to SIDs from source addresses outside the SR Domain. This may be realized with the following

logic. Other methods with equivalent outcome are considered compliant:

* assign all interface addresses from prefix A/a

* at node k, all SIDs local to k are assigned from prefix Sk/sk

* configure each internal interface of each SR node k in the SR Domain with an inbound IACL which drops any incoming packet with a destination address in Sk/sk if the source address is not in A/a.

5.2. SR Domain as A Single System with Delegation Among Components

All intra SR Domain packets are of the SR Domain. The IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

All inter domain packets are encapsulated for the part of the packet journey that is within the SR Domain. The outer IPv6 header is originated by a node of the SR Domain, and is destined to a node of the SR Domain.

As a consequence, any packet within the SR Domain is of the SR Domain.

The SR Domain is a system in which the operator may want to distribute or delegate different operations of the outer most header to different nodes within the system.

An operator of an SR domain may choose to delegate SRH addition to a host node within the SR domain, and validation of the contents of any SRH to a more trusted router or switch attached to the host. Consider a top of rack switch (T) connected to host (H) via interface (I). H receives an SRH (SRH1) with a computed HMAC via some SDN method outside the scope of this document. H classifies traffic it sources and adds SRH1 to traffic requiring a specific SLA. T is configured with an IACL on I requiring verification of the SRH for any packet destined to the SID block of the SR Domain (S/s). T checks and verifies that SRH1 is valid, contains an HMAC TLV and verifies the HMAC.

An operator of the SR Domain may choose to have all segments in the SR Domain verify the HMAC. This mechanism would verify that the SRH segment list is not modified while traversing the SR Domain.

5.3. MTU Considerations

An SR Domain ingress edge node encapsulates packets traversing the SR Domain, and needs to consider the MTU of the SR Domain. Within the SR Domain, well known mitigation techniques are RECOMMENDED, such as deploying a greater MTU value within the SR Domain than at the ingress edges.

Encapsulation with an outer IPv6 header and SRH share the same MTU and fragmentation considerations as IPv6 tunnels described in [RFC2473]. Further investigation on the limitation of various tunneling methods (including IPv6 tunnels) are discussed in [I-D.ietf-intarea-tunnels] and SHOULD be considered by operators when considering MTU within the SR Domain.

5.4. ICMP Error Processing

ICMP error packets generated within the SR Domain are sent to source nodes within the SR Domain. The invoking packet in the ICMP error message may contain an SRH. Since the destination address of a packet with an SRH changes as each segment is processed, it may not be the destination used by the socket or application that generated the invoking packet.

For the source of an invoking packet to process the ICMP error message, the ultimate destination address of the IPv6 header may be required. The following logic is used to determine the destination address for use by protocol error handlers.

- o Walk all extension headers of the invoking IPv6 packet to the routing extension header preceding the upper layer header.
  - * If routing header is type TBD IANA (SRH)
    - + The SID at Segment List[0] may be used as the destination address of the invoking packet.

ICMP errors are then processed by upper layer transports as defined in [RFC4443].

For IP packets encapsulated in an outer IPv6 header, ICMP error handling is as defined in [RFC2473].

5.5. Load Balancing and ECMP

For any inter domain packet, the SR Source node MUST impose a flow label computed based on the inner packet. The computation of the flow label is as recommended in [RFC6438] for the sending Tunnel End Point.

For any intra domain packet, the SR Source node SHOULD impose a flow label computed as described in [RFC6437] to assist ECMP load balancing at transit nodes incapable of computing a 5-tuple beyond the SRH.

At any transit node within an SR domain, the flow label MUST be used as defined in [RFC6438] to calculate the ECMP hash toward the destination address. If flow label is not used, the transit node would likely hash all packets between a pair of SR Edge nodes to the same link.

At an SR segment endpoint node, the flow label MUST be used as defined in [RFC6438] to calculate any ECMP hash used to forward the processed packet to the next segment.

5.6. Other Deployments

Other deployment models and their implications on security, MTU, HMAC, ICMP error processing and interaction with other extension headers are outside the scope of this document.

6. Illustrations

This section provides illustrations of SRv6 packet processing at SR source, transit and SR segment endpoint nodes.

6.1. Abstract Representation of an SRH

For a node k, its IPv6 address is represented as Ak, its SRv6 SID is represented as Sk.

IPv6 headers are represented as the tuple of (source, destination). For example, a packet with source address A1 and destination address A2 is represented as (A1,A2). The payload of the packet is omitted.

An SR Policy is a list of segments. A list of segments is represented as <S1,S2,S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit.

(SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- o Source Address is SA, Destination Addresses is DA, and next-header is SRH.

- o SRH with SID list <S1, S2, S3> with SegmentsLeft = SL.

- Note the difference between the <> and () symbols. <S1, S2, S3> represents a SID list where the leftmost segment is the first segment. Whereas, (S3, S2, S1; SL) represents the same SID list but encoded in the SRH Segment List format where the leftmost segment is the last segment. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of detailed behavior, the (S3, S2, S1; SL) notation is more convenient.

At its SR Policy headend, the Segment List <S1,S2,S3> results in SRH (S3,S2,S1; SL=2) represented fully as:

```
    Segments Left=2
    Last Entry=2
    Flags=0
    Tag=0
    Segment List[0]=S3
    Segment List[1]=S2
    Segment List[2]=S1
```

6.2. Example Topology

The following topology is used in examples below:

```
           + * * * * * * * * * * * * * * * * * * * * +

           *          [8]                 [9]          *
                       |                   |
           *           |                   |           *
    [1]----[3]--------[5]----------------[6]---------[4]---[2]
           *           |                   |           *
                       |                   |
           *           |                   |           *
                       +--------[7]-------+
           *                                           *

           + * * * * * * * *  SR Domain  * * * * * * * * +
```

Figure 3

- 3 and 4 are SR Domain edge routers

- 5, 6, and 7 are all SR Domain routers

- 8 and 9 are hosts within the SR Domain

- 1 and 2 are hosts outside the SR Domain

- o The SR domain implements ingress filtering as per Section 5.1 and no external packet can enter the domain with a destination address equal to a segment of the domain.

6.3. Source SR Node

6.3.1. Intra SR Domain Packet

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> the packet is

```
P1: (A8,S7)(A9,S7; SL=1)
```

6.3.1.1. Reduced Variant

When host 8 sends a packet to host 9 via an SR Policy <S7,A9> and it wants to use a reduced SRH, the packet is

```
P2: (A8,S7)(A9; SL=1)
```

6.3.2. Inter SR Domain Packet - Transit

When host 1 sends a packet to host 2, the packet is

```
P3: (A1,A2)
```

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. Router 3 encapsulates the received packet P3 in an outer header with an SRH. The packet is

```
P4: (A3, S7)(S4, S7; SL=1)(A1, A2)
```

If the SR Policy contains only one segment (the egress router 4), the ingress Router 3 encapsulates P3 into an outer header (A3, S4) without SRH. The packet is

```
P5: (A3, S4)(A1, A2)
```

6.3.2.1. Reduced Variant

The SR Domain ingress router 3 receives P3 and steers it to SR Domain egress router 4 via an SR Policy <S7, S4>. If router 3 wants to use a reduced SRH, Router 3 encapsulates the received packet P3 in an outer header with a reduced SRH. The packet is

```
P6: (A3, S7)(S4; SL=1)(A1, A2)
```

6.3.3. Inter SR Domain Packet - Internal to External

When host 8 sends a packet to host 1, the packet is encapsulated for the portion of its journey within the SR Domain. From 8 to 3 the packet is

P7: (A8,S3)(A8,A1)

In the opposite direction, the packet generated from 1 to 8 is

P8: (A1,A8)

At node 3 P8 is encapsulated for the portion of its journey within the SR domain, with the outer header destined to segment S8. Resulting in

P9: (A3,S8)(A1,A8)

At node 8 the outer IPv6 header is removed by S8 processing, then processed again when received by A8.

6.4. Transit Node

Nodes 5 acts as transit nodes for packet P1, and sends packet

P1: (A8,S7)(A9,S7;SL=1)

on the interface toward node 7.

6.5. SR Segment Endpoint Node

Node 7 receives packet P1 and, using the logic in Section 4.3.1, sends packet

P7: (A8,A9)(A9,S7; SL=0)

on the interface toward router 6.

6.6. Delegation of Function with HMAC Verification

This section describes how a function may be delegated within the SR Domain. In the following sections consider a host 8 connected to a top of rack 5.

6.6.1. SID List Verification

An operator may prefer to apply the SRH at source 8, while 5 verifies the SID list is valid.

For illustration purpose, an SDN controller provides 8 an SRH terminating at node 9, with segment list <S5,S7,S6,A9>, and HMAC TLV computed for the SRH. The HMAC key ID and key associated with the HMAC TLV is shared with 5. Node 8 does not know the key. Node 5 is configured with an IACL applied to the interface connected to 8, requiring HMAC verification for any packet destined to S/s.

Node 8 originates packets with the received SRH including HMAC TLV.

P15:(A8,S5)(A9,S6,S7,S5;SL=3;HMAC)

Node 5 receives and verifies the HMAC for the SRH, then forwards the packet to the next segment

P16:(A8,S7)(A9,S6,S7,S5;SL=2;HMAC)

Node 6 receives

P17:(A8,S6)(A9,S6,S7,S5;SL=1;HMAC)

Node 9 receives

P18:(A8,A9)(A9,S6,S7,S5;SL=0;HMAC)

This use of an HMAC is particularly valuable within an enterprise based SR Domain [SRN].

7. Security Considerations

This section reviews security considerations related to the SRH, given the SRH processing and deployment models discussed in this document.

As described in Section 5, it is necessary to filter packets ingress to the SR Domain, destined to SIDs within the SR Domain (i.e., bearing a SID in the destination address). This ingress filtering is via an IACL at SR Domain ingress border nodes. Additional protection is applied via an IACL at each SR Segment Endpoint node, filtering packets not from within the SR Domain, destined to SIDs in the SR Domain. ACLs are easily supported for small numbers of prefixes, making summarization important, and when the prefixes requiring filtering is kept to a seldom changing set.

Additionally, ingress filtering of IPv6 source addresses as recommended in BCP38 [RFC2827] SHOULD be used.

7.1. Source Routing Attacks

An SR domain implements distributed and per node protection as described in section 5.1. Additionally, domains deny traffic with spoofed addresses by implementing the recommendations in BCP 84 [RFC3704].

Full implementation of the recommended protection blocks the attacks documented in [RFC5095] from outside the SR domain, including bypassing filtering devices, reaching otherwise unreachable Internet systems, network topology discovery, bandwidth exhaustion, and defeating anycast.

Failure to implement distributed and per node protection allows attackers to bypass filtering devices and exposes the SR Domain to these attacks.

Compromised nodes within the SR Domain may mount the attacks listed above along with other known attacks on IP networks (e.g. DOS/DDOS, topology discovery, man-in-the-middle, traffic interception/ siphoning).

7.2. Service Theft

Service theft is defined as the use of a service offered by the SR Domain by a node not authorized to use the service.

Service theft is not a concern within the SR Domain as all SR Source nodes and SR segment endpoint nodes within the domain are able to utilize the services of the Domain. If a node outside the SR Domain learns of segments or a topological service within the SR domain, IACL filtering denies access to those segments.

7.3. Topology Disclosure

The SRH is unencrypted and may contain SIDs of some intermediate SR-nodes in the path towards the destination within the SR Domain. If packets can be snooped within the SR Domain, the SRH may reveal topology, traffic flows, and service usage.

This is applicable within an SR Domain, but the disclosure is less relevant as an attacker has other means of learning topology, flows, and service usage.

7.4. ICMP Generation

The generation of ICMPv6 error messages may be used to attempt denial-of-service attacks by sending an error-causing destination address or SRH in back-to-back packets. An implementation that correctly follows Section 2.4 of [RFC4443] would be protected by the ICMPv6 rate-limiting mechanism.

7.5. Applicability of AH

The SR Domain is a trusted domain, as defined in [RFC8402] Section 2 and Section 8.2. The SR Source is trusted to add an SRH (optionally verified as having been generated by a trusted source via the HMAC TLV in this document), and segments advertised within the domain are trusted to be accurate and advertised by trusted sources via a secure control plane. As such the SR Domain does not rely on the Authentication Header (AH) as defined in [RFC4302] to secure the SRH.

The use of SRH with AH by an SR source node, and processing at a SR segment endpoint node is not defined in this document. Future documents may define use of SRH with AH and its processing.

8. IANA Considerations

This document makes the following registrations in the Internet Protocol Version 6 (IPv6) Parameters "Routing Type" registry maintained by IANA:

Suggested Value	Description	Reference
4	Segment Routing Header (SRH)	This document

This document makes the following registrations in "Type 4 - Parameter Problem" message of the "Internet Control Message Protocol version 6 (ICMPv6) Parameters" registry maintained by IANA:

CODE	NAME/DESCRIPTION
TBD IANA	SR Upper-layer Header Error

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the SRH, in accordance with BCP 26, [RFC8126].

The following terms are used here with the meanings defined in BCP 26: "namespace", "assigned value", "registration".

The following policies are used here with the meanings defined in BCP 26 [RFC8126]: "IETF Review".

8.1. Segment Routing Header Flags Registry

This document requests the creation of a new IANA managed registry to identify SRH Flags Bits. The registration procedure is "IETF Review". Suggested registry name is "Segment Routing Header Flags". Flags is 8 bits.

8.2. Segment Routing Header TLVs Registry

This document requests the creation of a new IANA managed registry to identify SRH TLVs. The registration procedure is "IETF Review". Suggested registry name is "Segment Routing Header TLVs". A TLV is identified through an unsigned 8 bit codepoint value, with assigned values 0-127 for TLVs that do not change en route, and 128-255 for TLVs that may change en route. The following codepoints are defined in this document:

Assigned Value	Description	Reference
0	Pad1 TLV	This document
1	Reserved	This document
2	Reserved	This document
3	Reserved	This document
4	PadN TLV	This document
5	HMAC TLV	This document
6	Reserved	This document
124-126	Experimentation and Test	This document
127	Reserved	This document
252-254	Experimentation and Test	This document
255	Reserved	This document

Values 1,2,3,6 were defined in draft versions of this specification and are Reserved for backwards compatibility with early implementations and should not be reassigned. Values 127 and 255 are Reserved to allow for expansion of the Type field in future specifications if needed.

9. Implementation Status

This section is to be removed prior to publishing as an RFC.

See [I-D.matsushima-spring-srv6-deployment-status] for updated deployment and interoperability reports.

9.1. Linux

Name: Linux Kernel v4.14

Status: Production

Implementation: adds SRH, performs END processing, supports HMAC TLV

Details: https://irtf.org/anrw/2017/anrw17-final3.pdf and [I-D.filsfils-spring-srv6-interop]

9.2. Cisco Systems

Name: IOS XR and IOS XE

Status: Production (IOS XR), Pre-production (IOS XE)

Implementation: adds SRH, performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

9.3. FD.io

Name: VPP/Segment Routing for IPv6

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

Details: https://wiki.fd.io/view/VPP/Segment_Routing_for_IPv6 and [I-D.filsfils-spring-srv6-interop]

9.4. Barefoot

Name: Barefoot Networks Tofino NPU

Status: Prototype

Implementation: performs END processing, no TLV processing

Details: [I-D.filsfils-spring-srv6-interop]

9.5. Juniper

Name: Juniper Networks Trio and vTrio NPU’s

Status: Prototype & Experimental

Implementation: SRH insertion mode, Process SID where SID is an interface address, no TLV processing

9.6. Huawei

Name: Huawei Systems VRP Platform

Status: Production

Implementation: adds SRH, performs END processing, no TLV processing

10. Contributors

Kamran Raza, Zafar Ali, Brian Field, Daniel Bernier, Ida Leung, Jen Linkova, Ebben Aries, Tomoya Kosugi, Eric Vyncke, David Lebrun, Dirk Steinberg, Robert Raszuk, Dave Barach, John Brzozowski, Pierre Francois, Nagendra Kumar, Mark Townsley, Christian Martin, Roberta Maglione, James Connolly, Aloys Augustin contributed to the content of this document.

11. Acknowledgements

The authors would like to thank Ole Troan, Bob Hinden, Ron Bonica, Fred Baker, Brian Carpenter, Alexandru Petrescu, Punit Kumar Jaiswal, David Lebrun, Benjamin Kaduk, Frank Xialiang, Mirja Kuhlewind, Roman Danyliw, Joe Touch, and Magnus Westerlund for their comments to this document.

12. References

12.1. Normative References

[FIPS180-4] National Institute of Standards and Technology, "FIPS 180-4 Secure Hash Standard (SHS)", March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <https://www.rfc-editor.org/info/rfc2104>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/info/rfc2827>.

[RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, <https://www.rfc-editor.org/info/rfc3704>.

[RFC4107] Bellovin, S. and R. Housley, "Guidelines for Cryptographic Key Management", BCP 107, RFC 4107, DOI 10.17487/RFC4107, June 2005, <https://www.rfc-editor.org/info/rfc4107>.

[RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <https://www.rfc-editor.org/info/rfc4302>.

[RFC5095] Abley, J., Savola, P., and G. Neville-Neil, "Deprecation of Type 0 Routing Headers in IPv6", RFC 5095, DOI 10.17487/RFC5095, December 2007, <https://www.rfc-editor.org/info/rfc5095>.

[RFC6407] Weis, B., Rowles, S., and T. Hardjono, "The Group Domain of Interpretation", RFC 6407, DOI 10.17487/RFC6407, October 2011, <https://www.rfc-editor.org/info/rfc6407>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <https://www.rfc-editor.org/info/rfc6437>.

[RFC6438] Carpenter, B. and S. Amante, "Using the IPv6 Flow Label for Equal Cost Multipath Routing and Link Aggregation in Tunnels", RFC 6438, DOI 10.17487/RFC6438, November 2011, <https://www.rfc-editor.org/info/rfc6438>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

12.2. Informative References

[I-D.filsfils-spring-srv6-interop]
Filsfils, C., Clad, F., Camarillo, P., Abdelsalam, A., Salsano, S., Bonaventure, O., Horn, J., and J. Liste, "SRv6 interoperability report", draft-filsfils-spring-srv6-interop-02 (work in progress), March 2019.

[I-D.ietf-intarea-tunnels]
Touch, J. and M. Townsley, "IP Tunnels in the Internet Architecture", draft-ietf-intarea-tunnels-10 (work in progress), September 2019.

[I-D.matsushima-spring-srv6-deployment-status]
Matsushima, S., Filsfils, C., Ali, Z., and Z. Li, "SRv6 Implementation and Deployment Status", draft-matsushima-spring-srv6-deployment-status-02 (work in progress), October 2019.

[RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", STD 89, RFC 4443, DOI 10.17487/RFC4443, March 2006, <https://www.rfc-editor.org/info/rfc4443>.

[RFC5308] Hopps, C., "Routing IPv6 with IS-IS", RFC 5308, DOI 10.17487/RFC5308, October 2008, <https://www.rfc-editor.org/info/rfc5308>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <https://www.rfc-editor.org/info/rfc5340>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[SRN] Lebrun, D., Jadin, M., Clad, F., Filsfils, C., and O. Bonaventure, "Software Resolved Networks: Rethinking Enterprise Networks with IPv6 Segment Routing", 2018, <https://inl.info.ucl.ac.be/system/files/sosr18-final15-embedfonts.pdf>.

Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Brussels
BE

Email: cfilsfil@cisco.com

Darren Dukes (editor)
Cisco Systems, Inc.
Ottawa
CA

Email: ddukes@cisco.com

Stefano Previdi
Huawei
Italy

Email: stefano@previdi.net

John Leddy
Individual
US

Email: john@leddy.net

Satoru Matsushima
Softbank

Email: satoru.matsushima@g.softbank.co.jp

Daniel Voyer
Bell Canada

Email: daniel.voyer@bell.ca

```
SPRING                                                  C. Filsfils, Ed.

Internet-Draft                                         P. Camarillo, Ed.
Intended status: Standards Track                     Cisco Systems, Inc.
Expires: June 21, 2020                                         J. Leddy
                                                 Individual Contributor
                                                               D. Voyer
                                                            Bell Canada
                                                           S. Matsushima
                                                               SoftBank
                                                                  Z. Li
                                                    Huawei Technologies
                                                      December 19, 2019
```

SRv6 Network Programming
draft-ietf-spring-srv6-network-programming-07

Abstract

The SRv6 Network Programming framework enables a network operator or an application to specify a packet packet processing program by encoding a sequence of instructions in the IPv6 packet header.

Each instruction is implemented on one or several nodes in the network and identified by an SRv6 Segment Identifier in the packet.

This document defines the SRv6 Network Programming concept and specifies the base set of SRv6 behaviors that enables the creation of interoperable overlays with underlay optimization (Service Level Agreements).

Status of This Memo

Table of Contents

1. Introduction

Segment Routing [RFC8402] leverages the source routing paradigm. An ingress node steers a packet through an ordered list of instructions, called segments. Each one of these instructions represents a function to be called at a specific location in the network. A function is locally defined on the node where it is executed and may range from simply moving forward in the segment list to any complex user-defined behavior. Network programming combines segment routing functions, both simple and complex, to achieve a networking objective that goes beyond mere packet routing.

This document defines the SRv6 Network Programming concept and specifies the main segment routing behaviors to enable the creation of interoperable overlays with underlay optimization (Service Level Agreement).

The companion document [I-D.filsfils-spring-srv6-net-pgm-illustration] illustrates the concepts defined in this document.

Familiarity with the Segment Routing Header [I-D.ietf-6man-segment-routing-header] is expected.

2. Terminology

The following terms used within this document are defined in [RFC8402]: Segment Routing, SR Domain, Segment ID (SID), SRv6, SRv6 SID, Active Segment, SR Policy, Prefix SID and Adjacency SID.

The following terms used within this document are defined in [I-D.ietf-6man-segment-routing-header]: SRH, SR Source Node, Transit Node, SR Segment Endpoint Node and Reduced SRH.

NH: Next-header field of the IPv6 header[RFC8200]. NH=SRH means that the next-header of the IPv6 header is Routing Header for IPv6(43) with the Type field set to 4.

SL: The Segments Left field of the SRH

FIB: Forwarding Information Base. A FIB lookup is a lookup in the forwarding table.

SA: Source Address

DA: Destination Address

SRv6 SID function: The function part of the SID is an opaque identification of a local behavior bound to the SID. It is formally defined in Section 3.1 of this document.

SRv6 segment endpoint behavior: A packet processing behavior executed at an SRv6 segment endpoint. Section 4 of this document defines SRv6 segment endpoint behaviors related to traffic-engineering and overlay use-cases. Other behaviors (e.g. service programming) are outside the scope of this document.

An SR Policy is resolved to a SID list. A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SR path.

(SA,DA) (S3, S2, S1; SL) represents an IPv6 packet with:

- Source Address is SA, Destination Address is DA, and next-header is SRH

- SRH with SID list <S1, S2, S3> with Segments Left = SL

- Note the difference between the <> and () symbols: <S1, S2, S3> represents a SID list where S1 is the first SID and S3 is the last SID to traverse. (S3, S2, S1; SL) represents the same SID list but encoded in the SRH format where the rightmost SID in the SRH is the first SID and the leftmost SID in the SRH is the last SID. When referring to an SR policy in a high-level use-case, it is simpler to use the <S1, S2, S3> notation. When referring to an illustration of the detailed packet behavior, the (S3, S2, S1; SL) notation is more convenient.

- The payload of the packet is omitted.

SRH[n]: A shorter representation of Segment List[n], as defined in [I-D.ietf-6man-segment-routing-header].

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. SRv6 SID

RFC8402 defines an SRv6 Segment Identifier as an IPv6 address explicitly associated with the segment.

When an SRv6 SID is in the Destination Address field of an IPv6 header of a packet, it is routed through an IPv6 network as an IPv6 address.

Its processing is defined in [I-D.ietf-6man-segment-routing-header] section 4.3 and reproduced here as a reminder.

> Without constraining the details of an implementation, the SR segment endpoint node creates Forwarding Information Base (FIB) entries for its local SIDs.
>
> When an SRv6-capable node receives an IPv6 packet, it performs a longest-prefix-match lookup on the packets destination address. This lookup can return any of the following:
>
> - A FIB entry that represents a locally instantiated SRv6 SID
>
> - A FIB entry that represents a local interface, not locally instantiated as an SRv6 SID
>
> - A FIB entry that represents a non-local route
>
> - No Match

This document formally defines behaviors and parameters for SRv6 SIDs.

3.1. SID Format

This document defines an SRv6 SID as consisting of LOC:FUNCT:ARG, where a locator (LOC) is encoded in the L most significant bits of the SID, followed by F bits of function (FUNCT) and A bits of arguments (ARG). L, the locator length, is flexible, and an operator is free to use the locator length of their choice. F and A may be any value as long as L+F+A <= 128. When L+F+A is less than 128 then the reminder of the SID MUST be zero.

A locator may be represented as B:N where B is the SRv6 SID block (IPv6 subnet allocated for SRv6 SIDs by the operator) and N is the identifier of the parent node instantiating the SID.

When the LOC part of the SRv6 SIDs is routable, it leads to the node which instantiates the SID.

The FUNCT is an opaque identification of a local behavior bound to the SID.

The term "function" refers to the bit-string in the SRv6 SID. The term "behavior" identifies the behavior bound to the SID. The behaviors are defined in Section 4 of this document.

An SRv6 endpoint behavior MAY require additional information for its processing (e.g. related to the flow or service). This information may be encoded in the ARG bits of the SID.

In such a case, the semantics and format of the ARG bits are defined as part of the SRv6 endpoint behavior specification.

The ARG value of a routed SID SHOULD remain constant among packets in a given flow. Varying ARG values among packets in a flow may result in different ECMP hashing and cause re-ordering.

3.2. SID Reachability

Most often, the node N would advertise IPv6 prefix(es) matching the LOC parts covering its SIDs or shorter-mask prefix. The distribution of these advertisements and calculation of their reachability are routing protocol specific aspects that are outside the scope of this document.

An SRv6 SID is said to be routed if its SID belongs to an IPv6 prefix advertised via a routing protocol. An SRv6 SID that does not fulfill this condition is non-routed.

Let’s provide a classic illustration:

Node N is configured explicitly with two SIDs: 2001:DB8:B:1:100:: and 2001:DB8:B:2:101::.

The network learns about a path to 2001:DB8:B:1::/64 via the IGP and hence a packet destined to 2001:DB8:B:1:100:: would be routed up to N. The network does not learn about a path to 2001:DB8:B:2::/64 via the IGP and hence a packet destined to 2001:DB8:B:2:101:: would not be routed up to N.

A packet could be steered to a non-routed SID 2001:DB8:B:2:101:: by using a SID list <...,2001:DB8:B:1:100::,2001:DB8:B:2:101::,...> where the non-routed SID is preceded by a routed SID to the same node. Routed and non-routed SRv6 SIDs are the SRv6 instantiation of global and local segments, respectively [RFC8402].

4. SR Endpoint Behaviors

Each FIB entry indicates the behavior associated with a SID instance and its parameters.

Following is a set of well-known behaviors that can be associated with a SID.

End	Endpoint function The SRv6 instantiation of a prefix SID [RFC8402]
End.X	Endpoint with Layer-3 cross-connect The SRv6 instantiation of a Adj SID [RFC8402]
End.T	Endpoint with specific IPv6 table lookup
End.DX6	Endpoint with decapsulation and IPv6 cross-connect e.g. IPv6-L3VPN (equivalent to per-CE VPN label)
End.DX4	Endpoint with decaps and IPv4 cross-connect e.g. IPv4-L3VPN (equivalent to per-CE VPN label)
End.DT6	Endpoint with decapsulation and IPv6 table lookup e.g. IPv6-L3VPN (equivalent to per-VRF VPN label)
End.DT4	Endpoint with decapsulation and IPv4 table lookup e.g. IPv4-L3VPN (equivalent to per-VRF VPN label)
End.DT46	Endpoint with decapsulation and IP table lookup e.g. IP-L3VPN (equivalent to per-VRF VPN label)
End.DX2	Endpoint with decapsulation and L2 cross-connect e.g. L2VPN use-case
End.DX2V	Endpoint with decaps and VLAN L2 table lookup e.g. EVPN Flexible cross-connect use-case
End.DT2U	Endpoint with decaps and unicast MAC L2table lookup e.g. EVPN Bridging unicast use-case
End.DT2M	Endpoint with decapsulation and L2 table flooding e.g. EVPN Bridging BUM use-case with ESI filtering
End.B6.Encaps	Endpoint bound to an SRv6 policy with encapsulation SRv6 instantiation of a Binding SID
End.B6.Encaps.RED	End.B6.Encaps with reduced SRH SRv6 instantiation of a Binding SID
End.BM	Endpoint bound to an SR-MPLS Policy SRv6 instantiation of an SR-MPLS Binding SID

The list is not exhaustive. In practice, any function can be attached to a local SID: e.g. a node N can bind a SID to a local VM or container which can apply any complex processing on the packet.

The following sub-sections detail the behaviors, introduced in this document, that a node (N) binds to a SID (S).

Section 4.16 defines flavors of some of these behaviors.

4.1. End: Endpoint

The Endpoint behavior ("End" for short) is the most basic behavior. It is the instantiation of a Prefix-SID [RFC8402].

When N receives a packet whose IPv6 DA is S and S is a local End SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.      Send an ICMP Parameter Problem message to the Source Address
             Code 4 (SR Upper-layer Header Error),
             Pointer set to the offset of the upper-layer header.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.      Send an ICMP Time Exceeded message to the Source Address,
             Code 0 (Hop limit exceeded in transit),
             Interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > Last Entry+1)) {
S10.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.

S11.   }
S12.   Decrement Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Update IPv6 DA with Segment List[Segments Left]
S15.   Submit the packet to the egress IPv6 FIB lookup and
          transmission to the new destination
S16. }
```

Notes:
The End behavior operates on the same FIB table (i.e. VRF, L3 relay id) associated to the packet. Hence the FIB lookup on line S15 is done in the same FIB table as the ingress interface.

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End SID, send an ICMP parameter problem message to the Source Address and discard the packet. Error code 4 (SR Upper-layer Header Error) and Pointer set to the offset of the upper-layer header.

4.2. End.X: Layer-3 Cross-Connect

The "Endpoint with cross-connect to an array of layer-3 adjacencies" behavior (End.X for short) is a variant of the End behavior.

It is the SRv6 instantiation of an Adjacency-SID [RFC8402] and it is required to express any traffic-engineering policy.

An instance of the End.X behavior is associated with a set, J, of one or more Layer-3 adjacencies.

When N receives a packet destined to S and S is a local End.X SID, the line S15 from the End processing is replaced by the following:

```
S15.   Submit the packet to the IPv6 module for transmission
          to the new destination via a member of J
```

Notes:
S15. If the set J contains several L3 adjacencies, then one element of the set is selected based on a hash of the packet’s header Section 6.2.

If a node N has 30 outgoing interfaces to 30 neighbors, usually the operator would explicitly instantiate 30 End.X SIDs at N: one per layer-3 adjacency to a neighbor. Potentially, more End.X could be explicitly defined (groups of layer-3 adjacencies to the same neighbor or to different neighbors).

Note that if N has an outgoing interface bundle I to a neighbor Q made of 10 member links, N may allocate up to 11 End.X local SIDs: one for the bundle(LAG) itself and then up to one for each Layer-2 member link.

When the End.X behavior is associated with a BGP Next-Hop, it is the SRv6 instantiation of the BGP Peering Segments [RFC8402].

4.3. End.T: Specific IPv6 Table Lookup

The "Endpoint with specific IPv6 table lookup" behavior (End.T for short) is a variant of the End behavior.

The End.T behavior is used for multi-table operation in the core. For this reason, an instance of the End.T behavior is associated with an IPv6 FIB table T.

When N receives a packet destined to S and S is a local End.T SID, the line S15 from the End processing is replaced by the following:

```
S15.1.   Set the packet’s associated FIB table to T
S15.2.   Submit the packet to the egress IPv6 FIB lookup and
           transmission to the new destination
```

4.4. End.DX6: Decapsulation and IPv6 Cross-Connect

The "Endpoint with decapsulation and cross-connect to an array of IPv6 adjacencies" behavior (End.DX6 for short) is a variant of the End.X behavior.

One of the applications of the End.DX6 behavior is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This is equivalent to the per-CE VPN label in MPLS [RFC4364].

The End.DX6 SID MUST be the last segment in a SR Policy, and it is associated with one or more L3 IPv6 adjacencies J.

When N receives a packet destined to S and S is a local End.DX6 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DX6 SID, the following is done:

```
S01. If (Upper-Layer Header type != 41) {
S02.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S03. }
S04. Remove the outer IPv6 Header with all its extension headers
S05. Forward the exposed IPv6 packet to the L3 adjacency J
```

Notes:
S01. 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers.
S05. If the End.DX6 SID is bound to an array of L3 adjacencies, then one entry of the array is selected based on the hash of the packet's header Section 6.2.

4.5. End.DX4: Decapsulation and IPv4 Cross-Connect

The "Endpoint with decapsulation and cross-connect to an array of IPv4 adjacencies" behavior (End.DX4 for short) is a variant of the End.X behavior.

One of the applications of the End.DX4 behavior is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is not required. This is equivalent to the per-CE VPN label in MPLS [RFC4364].

The End.DX4 SID MUST be the last segment in a SR Policy, and it is associated with one or more L3 IPv4 adjacencies J.

When N receives a packet destined to S and S is a local End.DX4 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DX4 SID, the following is done:

```
S01. If (Upper-Layer Header type != 4) {
S02.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S03. }
S04. Remove the outer IPv6 Header with all its extension headers
S05. Forward the exposed IPv4 packet to the L3 adjacency J
```

Notes:
S01. 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers
S05. If the End.DX4 SID is bound to an array of L3 adjacencies, then one entry of the array is selected based on the hash of the packet’s header Section 6.2.

4.6. End.DT6: Decapsulation and Specific IPv6 Table Lookup

The "Endpoint with decapsulation and specific IPv6 table lookup" behavior (End.DT6 for short) is a variant of the End.T behavior.

One of the applications of the End.DT6 behavior is the L3VPNv6 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This is equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT6 may be defined for the main IPv6 table in which case and End.DT6 supports the equivalent of an IPv6inIPv6 decapsulation (without VPN/tenant implication).

The End.DT6 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv6 FIB table T.

When N receives a packet destined to S and S is a local End.DT6 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DT6 SID, N does the following:

```
S01. If (Upper-Layer Header type != 41) {
S02.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S03. }
S04. Remove the outer IPv6 Header with all its extension headers
S05. Set the packet’s associated FIB table to T
S06. Submit the packet to the egress IPv6 FIB lookup and
        transmission to the new destination
```

4.7. End.DT4: Decapsulation and Specific IPv4 Table Lookup

The "Endpoint with decapsulation and specific IPv4 table lookup" behavior (End.DT4 for short) is a variant of the End behavior.

One of the applications of the End.DT4 behavior is the L3VPNv4 use-case where a FIB lookup in a specific tenant table at the egress PE is required. This is equivalent to the per-VRF VPN label in MPLS [RFC4364].

Note that an End.DT4 may be defined for the main IPv4 table in which case an End.DT4 supports the equivalent of an IPv4inIPv6 decapsulation (without VPN/tenant implication).

The End.DT4 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv4 FIB table T.

When N receives a packet destined to S and S is a local End.DT4 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DT4 SID, N does the following:

```
S01. If (Upper-Layer Header type != 4) {
S02.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S03. }
S04. Remove the outer IPv6 Header with all its extension headers
S05. Set the packet’s associated FIB table to T
S06. Submit the packet to the egress IPv4 FIB lookup and
        transmission to the new destination
```

4.8. End.DT46: Decapsulation and Specific IP Table Lookup

The "Endpoint with decapsulation and specific IP table lookup" behavior (End.DT46 for short) is a variant of the End.DT4 and End.DT6 behavior.

One of the applications of the End.DT46 behavior is the L3VPN use-case where a FIB lookup in a specific IP tenant table at the egress PE is required. This is equivalent to single per-VRF VPN label (for IPv4 and IPv6) in MPLS[RFC4364].

Note that an End.DT46 may be defined for the main IP table in which case an End.DT46 supports the equivalent of an IPinIPv6 decapsulation(without VPN/tenant implication).

The End.DT46 SID MUST be the last segment in a SR Policy, and a SID instance is associated with an IPv4 FIB table T4 and an IPv6 FIB table T6.

When N receives a packet destined to S and S is a local End.DT46 SID, N does the following processing:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DT46 SID, N does the following:

```
S01. If (Upper-layer Header type == 4) {
S02.    Remove the outer IPv6 Header with all its extension headers
S03.    Set the packet's associated FIB table to T4
S04.    Submit the packet to the egress IPv4 FIB lookup and
           transmission to the new destination
S05. } Else if (Upper-layer Header type == 41) {
S06.    Remove the outer IPv6 Header with all its extension headers
S07.    Set the packet's associated FIB table to T6
S08.    Submit the packet to the egress IPv6 FIB lookup and
           transmission to the new destination
S09. } Else {
S10.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S11. }
```

4.9. End.DX2: Decapsulation and L2 Cross-Connect

The "Endpoint with decapsulation and Layer-2 cross-connect to an outgoing L2 interface (OIF)" (End.DX2 for short) is a variant of the endpoint behavior.

One of the applications of the End.DX2 behavior is the L2VPN/ EVPN[RFC7432] VPWS use-case.

The End.DX2 SID MUST be the last segment in a SR Policy, and it is associated with one outgoing interface I.

When N receives a packet destined to S and S is a local End.DX2 SID, N does:

```
S01. When an SRH is processed {
S02.   If (Segments Left != 0) {
S03.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   Proceed to process the next header in the packet
S06. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.DX2 SID, the following is done:

```
S01. If (Upper-Layer Header type != TBD1) {
S02.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S03. }
S04. Remove the outer IPv6 Header with all its extension headers and
        forward the Ethernet frame to the OIF I.
```

Notes:
S04. An End.DX2 behavior could be customized to expect a specific IEEE header (e.g. VLAN tag) and rewrite the egress IEEE header before forwarding on the outgoing interface.

4.10. End.DX2V: Decapsulation and VLAN L2 Table Lookup

The "Endpoint with decapsulation and specific VLAN table lookup" behavior (End.DX2V for short) is a variant of the End.DX2 behavior.

One of the applications of the End.DX2V behavior is the EVPN Flexible cross-connect use-case. The End.DX2V behavior is used to perform a lookup of the Ethernet frame VLANs in a particular L2 table. Any SID instance of the End.DX2V behavior is associated with an L2 Table T.

When N receives a packet whose IPv6 DA is S and S is a local End.DX2 SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is modified as follows:

```
S04. Remove the outer IPv6 Header with all its extension headers,
        lookup the exposed VLANs in L2 table T, and forward
        via the matched table entry.
```

Notes:
An End.DX2V behavior could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

4.11. End.DT2U: Decapsulation and Unicast MAC L2 Table Lookup

The "Endpoint with decapsulation and specific unicast MAC L2 table lookup" behavior (End.DT2U for short) is a variant of the End behavior.

One of the applications of the End.DT2U behavior is the EVPN Bridging unicast. Any SID instance of the End.DT2U behavior is associated with an L2 Table T.

When N receives a packet whose IPv6 DA is S and S is a local End.DT2U SID, the processing is identical to the End.DX2 behavior except for the Upper-layer header processing which is as follows:

```
S01. If (Upper-Layer Header type != TBD1) {
S02.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S03. }
S04. Remove the IPv6 header and all its extension headers
S05. Learn the exposed MAC Source Address in L2 Table T
S06. Lookup the exposed MAC Destination Address in L2 Table T
S07. If (matched entry in T) {
S08.    Forward via the matched table T entry
S09. } Else {
S10.    Forward via all L2 OIFs entries in table T
S11. }
```

Notes:
S05. In EVPN, the learning of the exposed inner MAC SA is done via the control plane.

4.12. End.DT2M: Decapsulation and L2 Table Flooding

The "Endpoint with decapsulation and specific L2 table flooding" behavior (End.DT2M for short) is a variant of the End.DT2U behavior.

Two of the applications of the End.DT2M behavior are the EVPN Bridging BUM with ESI filtering and the EVPN ETREE use-cases.

Any SID instance of this behavior is associated with a L2 table T. Additionally the behavior MAY take an argument: "Arg.FE2". It is an argument specific to EVPN ESI filtering and EVPN-ETREE used to exclude specific OIF (or set of OIFs) from L2 table T flooding.

When N receives a packet whose IPv6 DA is S and S is a local End.DT2M SID, the processing is identical to the End.DT2M behavior except for the Upper-layer header processing which is as follows:

```
S01. If (Upper-Layer Header type != TBD1) {
S02.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S03. }
S04. Remove the IPv6 header and all its extension headers
S05. Learn the exposed inner MAC Source Address in L2 Table T
S06. Forward via all L2 OIFs excluding the one specified in Arg.FE2
```

Notes:
S05. In EVPN, the learning of the exposed inner MAC SA is done via control plane

Arg.FE2 is encoded in the SID as an (k*x)-bit value. These bits represent a list of up to k OIFs, each identified with an x-bit value. Values k and x are defined on a per End.DT2M SID basis. The interface identifier 0 indicates an empty entry in the interface list.

4.13. End.B6.Encaps: Endpoint Bound to an SRv6 Policy w/ Encaps

This is a variation of the End behavior.

One of its applications is to express scalable traffic-engineering policies across multiple domains. It is the one of the SRv6 instantiations of a Binding SID [RFC8402].

An End.B6.Encaps SID is never the last segment in a SID list. Any SID instantiation is associated with an SR Policy B and a source address A.

When N receives a packet whose IPv6 DA is S and S is a local End.B6.Encaps SID, does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.      Send an ICMP Parameter Problem message to the Source Address
             Code 4 (SR Upper-layer Header Error),
             Pointer set to the offset of the upper-layer header.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.       Send an ICMP Time Exceeded message to the Source Address,
             Code 0 (Hop limit exceeded in transit),
             Interrupt packet processing and discard the packet.
S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > (Last Entry+1)) {
S10.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.
S11.   }
S12.   Decrement Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Push a new IPv6 header with its own SRH containing B
S15.   Set the outer IPv6 SA to A
S16.   Set the outer IPv6 DA to the first SID of B
S17.   Set the outer PayloadLength, Traffic Class, FlowLabel and
          Next-Header fields
S18.   Submit the packet to the egress IPv6 FIB lookup and
          transmission to the new destination
S19. }
```

Notes:
S14. The SRH MAY be omitted when the SRv6 Policy B only contains one SID and there is no need to use any flag, tag or TLV.
S17. The Payload Length, Traffic Class and Next-Header fields are set as per [RFC2473]. The Flow Label is computed as per [RFC6437].

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.B6.Encaps SID, send an ICMP parameter problem message to the Source Address and discard the packet. Error code 4 (SR Upper-layer Header Error), Pointer set to the offset of the upper-layer header.

4.14. End.B6.Encaps.Red: End.B6.Encaps with Reduced SRH

This is an optimization of the End.B6.Encaps behavior.

End.B6.Encaps.Red reduces the size of the SRH by one SID by excluding the first SID in the SRH of the new IPv6 header. Thus the first segment is only placed in the IPv6 Destination Address of the new IPv6 header and the packet is forwarded according to it.

The SRH Last Entry field is set as defined in Section 4.1.1 of [I-D.ietf-6man-segment-routing-header].

The SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

4.15. End.BM: Endpoint Bound to an SR-MPLS Policy

The "Endpoint bound to an SR-MPLS Policy" is a variant of the End behavior.

The End.BM behavior is required to express scalable traffic-engineering policies across multiple domains where some domains support the MPLS instantiation of Segment Routing. This is an SRv6 instantiation of an SR-MPLS Binding SID [RFC8402].

An End.BM SID is never the last SID, and any SID instantiation is associated with an SR-MPLS Policy B.

When N receives a packet whose IPv6 DA is S and S is a local End.BM SID, does:

```
S01. When an SRH is processed {
S02.   If (Segments Left == 0) {
S03.      Send an ICMP Parameter Problem message to the Source Address
             Code 4 (SR Upper-layer Header Error),
             Pointer set to the offset of the upper-layer header.
             Interrupt packet processing and discard the packet.
S04.   }
S05.   If (IPv6 Hop Limit <= 1) {
S06.      Send an ICMP Time Exceeded message to the Source Address,
             Code 0 (Hop limit exceeded in transit),
             Interrupt packet processing and discard the packet.

S07.   }
S08.   max_LE = (Hdr Ext Len / 2) - 1
S09.   If ((Last Entry > max_LE) or (Segments Left > (Last Entry+1)) {
S10.      Send an ICMP Parameter Problem to the Source Address,
             Code 0 (Erroneous header field encountered),
             Pointer set to the Segments Left field.
             Interrupt packet processing and discard the packet.

S11.   }
S12.   Decrement Hop Limit by 1
S13.   Decrement Segments Left by 1
S14.   Push the MPLS label stack for B
S15.   Submit the packet to the MPLS engine for transmission to the
          topmost label.
S16. }
```

When processing the Upper-layer header of a packet matching a FIB entry locally instantiated as an SRv6 End.BM SID, send an ICMP parameter problem message to the Source Address and discard the packet. Error code 4 (SR Upper-layer Header Error), Pointer set to the offset of the upper-layer header.

4.16. Flavors

The PSP, USP and USD flavors are variants of the End, End.X and End.T behaviors. For each of these behaviors these flavors MAY be supported for a SID either individually or in combinations.

4.16.1. PSP: Penultimate Segment Pop of the SRH

The SRH processing of the End, End.X and End.T behaviors are modified: after the instruction "S14. Update IPv6 DA with Segment List[Segments Left]" is executed, the following instructions must be executed as well:

```
 S14.1.   If (Segments Left == 0) {
 S14.2.      Update the Next Header field in the preceding header to the
                Next Header value of the SRH
 S14.3.      Decrease the IPv6 header Payload Length by the Hdr Ext Len
                value of the SRH
 S14.4.      Remove the SRH from the IPv6 extension header chain
 S14.5.   }
```

4.16.2. USP: Ultimate Segment Pop of the SRH

The SRH processing of the End, End.X and End.T behaviors are modified: the instructions S02-S04 are substituted by the following ones:

```
 S02.     If (Segments Left == 0) {
 S03.1.      Update the Next Header field in the preceding header to the
                Next Header value of the SRH
 S03.2.      Decrease the IPv6 header Payload Length by the Hdr Ext Len
                value of the SRH
 S03.3.      Remove the SRH from the IPv6 extension header chain
 S03.4.      Proceed to process the next header in the packet
 S04.     }
```

4.16.3. USD: Ultimate Segment Decapsulation

The SRH processing of the End, End.X and End.T behaviors are modified: the instructions S02-S04 are substituted by the following ones:

```
   S02.   If (Segments Left == 0) {
   S03.      Skip the SRH processing and proceed to the next header
   S04.   }
```

Further on, the Upper-layer header processing of the End, End.X and End.T behaviors are modified as follows:

```
End:
S01. If (Upper-layer Header type == 41 || 4) {
S02.    Remove the outer IPv6 Header with all its extension headers
S03.    Submit the packet to the egress IP FIB lookup and
           transmission to the new destination
S04. } Else {
S05.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.

S06. }

End.T:
S01. If (Upper-layer Header type == 41 || 4) {
S02.    Remove the outer IPv6 Header with all its extension headers
S03.    Set the packet’s associated FIB table to T
S04.    Submit the packet to the egress IP FIB lookup and
           Transmission to the new destination
S05. } Else {
S06.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S07. }

End.X:
S01. If (Upper-layer Header type == 41 || 4) {
S02.    Remove the outer IPv6 Header with all its extension headers
S03.    Forward the exposed IP packet to the L3 adjacency J
S04. } Else {
S05.    Send an ICMP Parameter Problem message to the Source Address
           Code 4 (SR Upper-layer Header Error),
           Pointer set to the offset of the upper-layer header.
           Interrupt packet processing and discard the packet.
S06. }
```

An implementation that supports the USD flavor in conjunction with the USP flavor MAY optimize the packet processing by first looking whether the conditions for the USD flavor are met, in which case it can proceed with USD processing else do USP processing.

5. SR Policy Headend Behaviors

This section describes a set of SR Policy Headend behaviors.

H.Encaps	SR Headend Behavior with Encapsulation in an SR Policy
H.Encaps.Red	H.Encaps with Reduced Encapsulation
H.Encaps.L2	H.Encaps Applied to Received L2 Frames
H.Encaps.L2.Red	H.Encaps.Red Applied to Received L2 Frames

This list can be expanded in case any new functionality requires it.

5.1. H.Encaps: SR Headend with Encapsulation in an SRv6 Policy

Node N receives two packets P1=(A, B2) and P2=(A,B2)(B3, B2, B1; SL=1). B2 is neither a local address nor SID of N.

N steers the transit packets P1 and P2 into an SR Policy with a Source Address T and a Segment list <S1, S2, S3>.

The H.Encaps transit encapsulation behavior is defined as follows:

```
S01.   Push an IPv6 header with its own SRH (S3, S2, S1; SL=2)
S02.   Set outer IPv6 SA = T and outer IPv6 DA = S1
S03.   Set outer payload length, traffic class and flow label
S04.   Update the Next-Header value
S05.   Decrement inner Hop Limit or TTL
S06.   Submit the packet to the IPv6 module for transmission to S1
```

After the H.Encaps behavior, P1 and P2 respectively look like:

- (T, S1) (S3, S2, S1; SL=2) (A, B2)

- (T, S1) (S3, S2, S1; SL=2) (A, B2) (B3, B2, B1; SL=1)

The received packet is encapsulated unmodified (with the exception of the TTL or Hop Limit that is decremented as described in [RFC2473]).

The H.Encaps behavior is valid for any kind of Layer-3 traffic. This behavior is commonly used for L3VPN with IPv4 and IPv6 deployments. It may be also used for TI-LFA[I-D.ietf-rtgwg-segment-routing-ti-lfa] at the point of local repair.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

S03: As described in [RFC6437] (IPv6 Flow Label Specification)

5.2. H.Encaps.Red: H.Encaps with Reduced Encapsulation

The H.Encaps.Red behavior is an optimization of the H.Encaps behavior.

H.Encaps.Red reduces the length of the SRH by excluding the first SID in the SRH of the pushed IPv6 header. The first SID is only placed in the Destination Address field of the pushed IPv6 header.

After the H.Encaps.Red behavior, P1 and P2 respectively look like:

- (T, S1) (S3, S2; SL=2) (A, B2)

- (T, S1) (S3, S2; SL=2) (A, B2) (B3, B2, B1; SL=1)

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

5.3. H.Encaps.L2: H.Encaps Applied to Received L2 Frames

The H.Encaps.L2 behavior encapsulates a received Ethernet [Ethernet] frame and its attached VLAN header, if present, in an IPv6 packet with an SRH. The Ethernet frame becomes the payload of the new IPv6 packet.

The Next Header field of the SRH MUST be set to TBD1.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

The encapsulating node MUST remove the preamble or frame check sequence (FCS) from the Ethernet frame upon encapsulation and the decapsulating node MUST regenerate the preamble or FCS before forwarding Ethernet frame.

5.4. H.Encaps.L2.Red: H.Encaps.Red Applied to Received L2 frames

The H.Encaps.L2.Red behavior is an optimization of the H.Encaps.L2 behavior.

H.Encaps.L2.Red reduces the length of the SRH by excluding the first SID in teh SRH of the pushed IPv6 header. The first SID is only places in the Destination Address field of the pushed IPv6 header.

The push of the SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

6. Operation

6.1. Counters

Any SRv6 capable node SHOULD implement the following set of combined counters (packets and bytes):

- CNT-1: Per local SID entry, traffic that matched that SID and was processed correctly.

- CNT-2: Per SRv6 Policy, traffic steered into it and processed correctly.

Furthermore, an SRv6 capable node SHOULD maintain an aggregate counter CNT-3 tracking the IPv6 packets received with an IPv6 Destination Address matching a local interface address that is not a locally instantiated SID and containing an SRH with a Segments Left value different from 0.

6.2. Flow-based Hash Computation

When a flow-based selection within a set needs to be performed, the source address, the destination address and the flow label MUST be included in the flow-based hash.

This occurs when a FIB lookup is performed and multiple ECMP paths exist to the updated destination address.

This occurs when End.X, End.DX4, or End.DX6 are bound to an array of adjacencies.

This occurs when the packet is steered in an SR policy whose selected path has multiple SID lists.

Additionally, any transit router in an SRv6 domain includes the outer flow label in its ECMP load-balancing hash [RFC6437].

6.3. OAM

[I-D.ietf-6man-spring-srv6-oam] defines OAM behaviors for SRv6. This includes the definition of the SRH Flag ‘O-bit’, as well as additional SR Endpoint behaviors for OAM purposes.

7. Security Considerations

The security considerations for Segment Routing are discussed in [RFC8402]. More specifically for SRv6 the security considerations and the mechanisms for securing an SR domain are discussed in [I-D.ietf-6man-segment-routing-header]. Together, they describe the required security mechanisms that allow establishment of an SR domain of trust to operate SRv6-based services for internal traffic while preventing any external traffic from accessing or exploiting the SRv6-based services.

This document introduces SRv6 Endpoint and Transit Nodes behaviors for implementation on SRv6 capable nodes in the network. As such, this document does not introduce any new security considerations.

8. Control Plane

In an SDN environment, one expects the controller to explicitly provision the SIDs and/or discover them as part of a service discovery function. Applications residing on top of the controller could then discover the required SIDs and combine them to form a distributed network program.

The concept of "SRv6 network programming" refers to the capability for an application to encode any complex program as a set of individual functions distributed through the network. Some functions relate to underlay SLA, others to overlay/tenant, others to complex applications residing in VM and containers.

This section provides a high level overview of the control-plane protocols involved with SRv6 and their specification.

8.1. IGP

The End, End.T and End.X SIDs express topological behaviors and hence are expected to be signaled in the IGP together with the flavors PSP, USP and USD[I-D.ietf-lsr-isis-srv6-extensions]. The IGP also advertises the support for SRv6 capabilities of the node.

The presence of SIDs in the IGP do not imply any routing semantics to the addresses represented by these SIDs. The routing reachability to an IPv6 address is solely governed by the, non-SID-related, IGP prefix reachability information that includes locators. Routing is not governed neither influenced in any way by a SID advertisement in the IGP.

These SIDs provide important topological behaviors for the IGP to build TI-LFA[I-D.ietf-rtgwg-segment-routing-ti-lfa] based FRR solutions and for TE processes relying on IGP topology database to build SR policies.

8.2. BGP-LS

BGP-LS provides the functionality for topology discovery that includes the SRv6 capabilities of the nodes, their locators and locally instantiated SIDs [I-D.ietf-idr-bgpls-srv6-ext]. This enables controllers or applications to build an inter-domain topology that can be used for computation of SR Policies using the SRv6 SIDs.

8.3. BGP IP/VPN/EVPN

The End.DX4, End.DX6, End.DT4, End.DT6, End.DT46, End.DX2, End.DX2V, End.DT2U and End.DT2M SIDs can be signaled in BGP [I-D.ietf-bess-srv6-services].

8.4. Summary

The following table summarizes behaviors for SIDs that can be signaled in which each respective control plane protocol.

	IGP	BGP-LS	BGP IP/VPN/EVPN
End (PSP, USP, USD)	X	X	
End.X (PSP, USP, USD)	X	X	
End.T (PSP, USP, USD)	X	X	
End.DX6	X	X	X
End.DX4	X	X	X
End.DT6	X	X	X
End.DT4	X	X	X
End.DT46	X	X	X
End.DX2		X	X
End.DX2V		X	X
End.DT2U		X	X
End.DT2M		X	X
End.B6.Encaps		X	
End.B6.Encaps.Red		X	
End.B6.BM		X	

Table 1: SRv6 locally instantiated SIDs signaling

The following table summarizes which transit capabilities are signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN/EVPN
H.Encaps	X	X	
H.Encaps.Red	X	X	
H.Encaps.L2		X	
H.Encaps.L2.Red		X	

Table 2: SRv6 transit behaviors signaling

The previous table describes generic capabilities. It does not describe specific instantiated SR policies.

For example, a BGP-LS advertisement of H.Encaps behavior would describe the capability of node N to perform a H.Encaps behavior, specifically it would describe how many SIDs could be pushed by N without significant performance degradation.

As a reminder, an SR policy is always assigned a Binding SID [RFC8402]. BSIDs are also advertised in BGP-LS as shown in Table 1. Hence, the Table 2 only focuses on the generic capabilities related to H.Encaps.

9. IANA Considerations

9.1. Ethernet Next Header Type

This document requests IANA to allocate, in the "Protocol Numbers" registry (https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml), a new value for "Ethernet" with the following definition: The value TBD1 in the Next Header field of an IPv6 header or any extension header indicates that the payload is an Ethernet [Ethernet].

9.2. SRv6 Endpoint Behaviors Registry

This document requests IANA to create a new top-level registry called "Segment Routing Parameters". This registry is being defined to serve as a top-level registry for keeping all other Segment Routing sub-registries.

Additionally, a new sub-registry "SRv6 Endpoint Behaviors" is to be created under top-level "Segment Routing Parameters" registry. This sub-registry maintains 16-bit identifiers for the SRv6 Endpoint behaviors. The range of the registry is 0-65535 (0x0000 - 0xFFFF) and has the following registration rules and allocation policies:

Range	Hex	Registration procedure	Notes
0	0x0000	Reserved	Invalid
1-32767	0x0001-0x7FFF	FCFS	
32768-65534	0x8000-0xFFFE	Reserved. Not to be allocated.	
65535	0xFFFF	Reserved	Opaque

Table 3: SRv6 Endpoint Behaviors Registry

9.2.1. Initial Registrations

The initial registrations for the sub-registry are as follows:

Value	Hex	Endpoint behavior	Reference
0	0x0000	Invalid	[This.ID]
1	0x0001	End (no PSP, no USP)	[This.ID]
2	0x0002	End with PSP	[This.ID]
3	0x0003	End with USP	[This.ID]
4	0x0004	End with PSP&USP	[This.ID]
5	0x0005	End.X (no PSP, no USP)	[This.ID]
6	0x0006	End.X with PSP	[This.ID]
7	0x0007	End.X with USP	[This.ID]
8	0x0008	End.X with PSP&USP	[This.ID]
9	0x0009	End.T (no PSP, no USP)	[This.ID]
10	0x000A	End.T with PSP	[This.ID]
11	0x000B	End.T with USP	[This.ID]
12	0x000C	End.T with PSP&USP	[This.ID]
13	0x000D	Reserved	-
14	0x000E	End.B6.Encaps	[This.ID]
15	0x000F	End.BM	[This.ID]
16	0x0010	End.DX6	[This.ID]
17	0x0011	End.DX4	[This.ID]
18	0x0012	End.DT6	[This.ID]
19	0x0013	End.DT4	[This.ID]
20	0x0014	End.DT46	[This.ID]
21	0x0015	End.DX2	[This.ID]
22	0x0016	End.DX2V	[This.ID]
23	0x0017	End.DT2U	[This.ID]
24	0x0018	End.DT2M	[This.ID]
25	0x0019	Reserved	[This.ID]
26	0x001A	Reserved	-

27	0x001B	End.B6.Encaps.Red	[This.ID]
28	0x001C	End with USD	[This.ID]
29	0x001D	End with PSP&USD	[This.ID]
30	0x001E	End with USP&USD	[This.ID]
31	0x001F	End with PSP, USP & USD	[This.ID]
32	0x0020	End.X with USD	[This.ID]
33	0x0021	End.X with PSP&USD	[This.ID]
34	0x0022	End.X with USP&USD	[This.ID]
35	0x0023	End.X with PSP, USP & USD	[This.ID]
36	0x0024	End.T with USD	[This.ID]
37	0x0025	End.T with PSP&USD	[This.ID]
38	0x0026	End.T with USP&USD	[This.ID]
39	0x0027	End.T with PSP, USP & USD	[This.ID]
40-32767		Unassigned	
32768-65534		Reserved	Change control under IETF
65535	0xFFFF	Opaque	[This.ID]

Table 4: IETF - SRv6 Endpoint Behaviors

Requests for allocation from within the FCFS range must include a point of contact and preferably also a brief description of how the value will be used. This information may be provided with a reference to an Internet Draft or an RFC or in some other documentation that is permanently and readily available.

10. Acknowledgements

The authors would like to acknowledge Stefano Previdi, Dave Barach, Mark Townsley, Peter Psenak, Thierry Couture, Kris Michielsen, Paul Wells, Robert Hanzl, Dan Ye, Gaurav Dawra, Faisal Iqbal, Jaganbabu Rajamanickam, David Toscano, Asif Islam, Jianda Liu, Yunpeng Zhang, Jiaoming Li, Narendra A.K, Mike Mc Gourty, Bhupendra Yadav, Sherif Toulan, Satish Damodaran, John Bettink, Kishore Nandyala Veera Venk, Jisu Bhattacharya and Saleem Hafeez.

11. Contributors

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Dirk Steinberg
Lapishills Consulting Limited
Cyprus

Email: dirk@lapishills.com

Robert Raszuk
Bloomberg LP
United States of America

Email: robert@raszuk.net

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Belgium

Email: bart.peirens@proximus.com

Hani Elmalky
Google
United States of America

Email: helmalky@google.com

Prem Jonnalagadda
Barefoot Networks
United States of America

Email: prem@barefootnetworks.com

Milad Sharif
Barefoot Networks
United States of America

Email: msharif@barefootnetworks.com

David Lebrun
Google
Belgium

Email: dlebrun@google.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Ahmed AbdelSalam
Gran Sasso Science Institute
Italy

Email: ahmed.abdelsalam@gssi.it

Gaurav Naik
Drexel University
United States of America

Email: gn@drexel.edu

Arthi Ayyangar
Arrcus, Inc
United States of America

Email: arthi@arrcus.com

Satish Mynam
Arrcus, Inc
United States of America

Email: satishm@arrcus.com

Wim Henderickx
Nokia
Belgium

Email: wim.henderickx@nokia.com

Shaowen Ma
Juniper
Singapore

Email: mashao@juniper.net

Ahmed Bashandy
Individual
United States of America

Email: abashandy.ietf@gmail.com

Francois Clad
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Kamran Raza
Cisco Systems, Inc.
Canada

Email: skraza@cisco.com

Darren Dukes
Cisco Systems, Inc.
Canada

Email: ddukes@cisco.com

Patrice Brissete
Cisco Systems, Inc.
Canada

Email: pbrisset@cisco.com

Zafar Ali
Cisco Systems, Inc.
United States of America

Email: zali@cisco.com

Ketan Talaulikar
Cisco Systems, Inc.
India

Email: ketant@cisco.com

12. References

12.1. Normative References

[Ethernet]
DigitalEquipment, Intel, and Xerox, "The Ethernet -- A Local Area Network: Data Link Layer and Physical Layer (Version 2.0)", November 1982.

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-26 (work in progress), October 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <https://www.rfc-editor.org/info/rfc2473>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

12.2. Informative References

[I-D.filsfils-spring-srv6-net-pgm-illustration]
Filsfils, C., Camarillo, P., Li, Z., Matsushima, S., Decraene, B., Steinberg, D., Lebrun, D., Raszuk, R., and J. Leddy, "Illustrations for SRv6 Network Programming", draft-filsfils-spring-srv6-net-pgm-illustration-01 (work in progress), August 2019.

[I-D.ietf-6man-spring-srv6-oam]
Ali, Z., Filsfils, C., Matsushima, S., Voyer, D., and M. Chen, "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ietf-6man-spring-srv6-oam-02 (work in progress), November 2019.

```
   [I-D.ietf-bess-srv6-services]
              Dawra, G., Filsfils, C., Raszuk, R., Decraene, B., Zhuang,
              S., and J. Rabadan, "SRv6 BGP based Overlay services",
              draft-ietf-bess-srv6-services-01 (work in progress),
              November 2019.

   [I-D.ietf-idr-bgpls-srv6-ext]
              Dawra, G., Filsfils, C., Talaulikar, K., Chen, M.,
              daniel.bernier@bell.ca, d., and B. Decraene, "BGP Link
              State Extensions for SRv6", draft-ietf-idr-bgpls-
              srv6-ext-01 (work in progress), July 2019.

   [I-D.ietf-lsr-isis-srv6-extensions]
              Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and
              Z. Hu, "IS-IS Extension to Support Segment Routing over
              IPv6 Dataplane", draft-ietf-lsr-isis-srv6-extensions-03
              (work in progress), October 2019.

   [I-D.ietf-rtgwg-segment-routing-ti-lfa]
              Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B.,
              Francois, P., Voyer, D., Clad, F., and P. Camarillo,
              "Topology Independent Fast Reroute using Segment Routing",
              draft-ietf-rtgwg-segment-routing-ti-lfa-01 (work in
              progress), March 2019.

   [RFC4364]  Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private
              Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February
              2006, <https://www.rfc-editor.org/info/rfc4364>.

   [RFC6437]  Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme,
              "IPv6 Flow Label Specification", RFC 6437,
              DOI 10.17487/RFC6437, November 2011,
              <https://www.rfc-editor.org/info/rfc6437>.

   [RFC7432]  Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A.,
              Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based
              Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February
              2015, <https://www.rfc-editor.org/info/rfc7432>.
```

Authors' Addresses

Clarence Filsfils (editor)
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

Pablo Camarillo Garvia (editor)
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com

John Leddy
Individual Contributor
United States of America

Email: john@leddy.net

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Satoru Matsushima
SoftBank
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7322
Japan

Email: satoru.matsushima@g.softbank.co.jp

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

ANNEX 4 – CISCO SR TEAM

Many thanks to such a great team.

Aaron Itskovich	Abbas Jafferjee	Abdel Baig
Abdeli-Abdelouahed Brimelli	Abdul Rehman	Abhay Roy
Abhijeet Padhye	Abhishek Gautam	Aboli Pimpalkhare
Acee Lindem (acee)	Ahmed Bashandy	Aisha Sanes
Al Rocheleau	Alan Xiao-rong Wang	Alex Tokar
Alexander Preusche	Ambrish Mehta	Amila Tharaperiya Gamage
Amin Qassoud -X	Amir Darehshoorzadeh	Amir Khan
Amit Agrawal	Amit Nigam	Amit Tamang
Amjad Pasha	Andrew Brown	Andrew Lyle
Andy Karch	Anjali Managoli	Anup Kumar Vasudevan
Apoorva Karan	Archana Rao	Arjun Sreekantiah
Arul Murugan	Arvindan Baskaran	Ashish Kumar
Asif Islam	Asif Islam	Avinash Prakash
Avinash Tadimalla	Avinash Tadimalla	Azam Matloob
Baalajee S	Balaji Ganesh	Balaji Karnam
Balaji Muthuvarathan	Bargav Sriperumbudur Thattai	Basavaraju Halappa
Benoit Godin	Bertrand Duvivier	Bhupendra Yadav
Bo Hu	Bo Wu	Boris Petrovic
Bruce McDougall	Carlos Pignataro	Carol Gal
Chran-Ham Chang	Chris Martin	Chuanfa Wang
Cindy Bai	Coleen Quadros	Dakshina Moorthy
Dan Ye	Danial Johari	Danial Johari
Darren Dukes	David Toscano	Deepak NyayachawadiAnanth
Deepti B	Dennis Cai	Devarajegowda M T
Dhanendra Jain	Dinuraj K	Divakaran Baskaran
Divyanshuman Divyanshuman	Dongling Duan	Elio Lerner
Eman Al Disi	Eric Fett	Eric Pruneau
Faisal Iqbal	Fan Kong	Fan Song
Federico Colasante	Fei Zhong	Francois Clad
Frank Fornoff	Frederic Trate	Gabor Solymar
Ganapathasaa Katwa	Gaurav Dawra	Geetanjalli Bhalla

Georgi Savov	Gowdemy Rajalingham	Gregory Brown
Guennoun Mouhcine	hadee Akhand	Hanane Tavasoli
Haowei Shi	Hareen Kancharla	Harish Kumar Chencharla Raghavendr
Hashim Abdullahi Sabriye	Hassan Sheikh	Hicham Ouahid
Howard Yang	Hubert Jin	Hui Mao
Ianik Semco	Imad Abdeljaouad	Jagadeesh Maiya
Jaganbabu Rajamanickam	James Pylakutty	Jayanth Srinivasa
Jeevani Patamse	Jeff Byzek	Jeff Williams
Jeffrey Chiang	Jiajian Yang	Jianda Liu
Jiaoming Li	Jing Lu	Jisu Bhattacharya
John Paul Varkey	Johnny Wang	Jon D Parker
Jose Liste	Joseph Chin	Juan Alcaide
Junaid Israr	Jurek Matuszewski	Kalai Sankaralingam
Kamran Raza	Kapil Zadoo	Kavya Ramachandra
Ketan Jivan Talaulikar	Khandaker Rafiul Hasan	Khanh Phan
Kiran Pillai	Kiran Srinivas	Kodanda Singamala
Kodandaramireddy Singamala	Kotta Hanumantha Rao	Kris Michielsen
Krishna Deevi	Krishna Muddenahally Ananthamurthy	L Srikanth
Lakshman Swaroop Babu	Laleh Sotoudeh	Lavanya Srivatsa
Leo Mermelstein	Les Ginsberg	Ling Zeng
Madhan Sankaranarayanan	Madhu Sharma	Malick Mohamed Usman
Manan Patel	Manav Bector	Manish Gupta
Manoj Abraham	Marek Karasek	Marianne Talaugon
Mark Vanheule	Mark Zheng	Masroor Vettu Parambil
Matt Gillies	Matt Hartley	Matthew Starkey
Mayur Shetty	Mengyao Ye	Mercia Zheng
Michael Cho	Michael MacKenzie	Michael Mc Gourty
Mike Koldychev	Mike Koldychev	Mike Mallin
Mike Taillon	Mike Taillon	Mingyang Sun
Misagh Tavanpour	Miya Kohno	Mohamad Ben Zeglam
Mohammed Mirza	Mouhcine Guennou	Mourad Atassi
Mudassar Rayani	Murali G Krishna	Murali Gandluru

Murthy Haresamudra	Naren Mididaddi	Narendra A.K
Narendra Sharma	Naresh Sharma	Navdeep Sondh
Nishant Dubey	Nitin Kumar	Pablo Camarillo
Patrice Brissette	Patrick Khordoc	Paul Wells
Payal Bhatia	Pengyan Zhang	Peter Huang
Peter Psenak	Pooja Doijode	Pooja Garla
Prabhu Vaithilingam	Prajeet G.C.	Prajeet GC
Pramod Eapen	Prasad S. Narasimha	Prashant Garimella
Prashanth Narsimhachar	Prathap Raju Hongere Devaraju	Pratima Kini
Praveena Sivaprakasam	Pritesh Mistry	Pritesh Mistry
Priya Murugesan	Radu Valceanu	Raghu Salotagi
Raghul Selvarajan	Rajasekar Raja	Rajasekhar Dande
Rajeev Gupta	Rajeev Mishra	Rajesh M. Venkateswaran
Rajesh Pandey	Rajesh Sharma	Rajiv Singh
Rajya Lakshmi Naralasetty	Rakesh Gandhi	Ram Dahal
Ramanaa H V	Ramesh Yakkala	Rathina Sabapathy Sabesan
Ravi Prakash	Ravi Rajarao	Ravindra Vaishampayan
Ray Kwok	Richard Vallee	Robert Adams
Robert Hanzl	Robert Sawaya	Roberta Maglione
Sadat Islam	Sagar Soni	Sahi Shah
Saleem Hafeez	Sam Kheirallah	Sameer Gulrajani
Samuel Sidor	Sankaralingam Thirunavukkarasu	Sankararaman Kasimuthuraman
Santanu Dasgupta	Santiago Freitas	Santosh Dalavaipatnam
Saravanan Arumugam	Satish Damodaran	Satya Mohanty
Saurabh Gupta	Savitha M	Sebastien Trottier
Senthilkumar Purushothaman	Seraph Li	Serge Krier
Sergey Plotnikov	Shahzad Yasin	Shankar Ramanathan
Sharanya Subramanian	Sharmila Palani	Sherif Toulan
Shishir Kumar	Shiv Kumar	Shyam Sethuram
Shyno Annie Jacob	Simardeep Ahuja	Sindhu Ramegowda
Siva Sivabalan	Sneha Bharadwaj	Songtao Chen
Soumik Bhattacharya	Sravani Kolli	Srejith Avikkal

Sridhar Santhanam	Srimanikandan Ganesan	Srini Ramabadran
Sriram Prasad	Sriram Rajaraman	Sriram Srinivasan
Stefan-Dragos Ipate	Stefano Previdi	Stephane Labetoulle
Steve Braaten	Steve Lee	Steven Luong
Sudheer Kalyanashetty	Sudhir Pandey	Suguna Ganti
Suguna Ganti	Sujit Chandrapati	Surjyendu Dhal
Sushek Shekar	Sushil Kumar	Susil Santhoshmull
Swadesh Agrawal	Swaraj Kumar Chikyala	Tarek Saad
Tarik Saleh	Thiyagarajan	Thomas Wang
Thuan Van Tran	Tiffany Cheng	Tim Laberge
Tony Tong	Umesh Dudani	Venkat Janakiram
Venkata Laxmi Sruthi	Venu Venugopal	Vibha V Uppin
Vibov Bhan	Victor Ji	Vijay Kachinthaya
Vinay Viswanath	Vinit Chanduka	Vinod Chandran
Vipul Shah	Viral Patel	Visweswar Rao Yechuri
Wanmathy Dolaastan	Wenji Zhao	Wentong Wang
Wes Beebee	Xianlei Du	Xiaobo Pi
Xiongbin Ma	Yao Zhao	YF Siu
Yikun Wang	Yoga Vetsa	Yong Wang
Youssef Al Sabbagh	Yue Gao	Yuri Tsier
Zafar Ali	Zhaoyang Yu	Zhenxia Zhang
Zhihao Hong	Zulfikar Ali Sahool Hameed	Zuoheng Qin