SPRING
Internet-Draft
Intended status: Standards Track
Expires: September 10, 2017

C. Fisfils
Cisco Systems, Inc.
J. Leddy
Comcast
D. Voyer
D. Bernier
Bell Canada
D. Steinberg
Steinberg Consulting
R. Raszuk
Bloomberg LP
S. Matsushima
SoftBank Telecom
D. Lebrun
Universite catholique de Louvain
B. Decraene
Orange
B. Peirens
Proximus
S. Salsano
Universita di Roma "Tor Vergata"
G. Naik
Drexel University
H. Elmalky
Ericsson
P. Jonnalagadda
M. Sharif
Barefoot Networks
A. Ayyangar
Arista
S. Mynam
Dell Force10 Networks
A. Bashandy
K. Raza
D. Dukes
F. Clad
P. Camarillo, Ed.
Cisco Systems, Inc.
March 9, 2017

SRv6 Network Programming

draft-filsfils-spring-srv6-network-programming-00

Abstract

This document describes the SRv6 network programming concept and its most basic functions.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 10, 2017.

Table of Contents

1. Introduction

This document defines the SRv6 network programming concept, its most frequent functions and the related terminology.

This document assumes familiarity with the Segment Routing Header [I-D.ietf-6man-segment-routing-header].

2. Terminology

SRH is the abbreviation for the Segment Routing Header. We assume that the SRH may be present multiple times inside each packet.

NH is the abbreviation of the IPv6 next-header field.

NH=SRH means that the next-header field is 43 with routing type 4.

When there are multiple SRHs, they must follow each other: the next-header field of all SRH except the last one must be SRH.

The effective next-header (ENH) is the next-header field of the IP header when no SRH is present, or is the next-header field of the last SRH.

In this version of the document, we assume that there is no other extension header than the SRH. These will be lifted in future versions of the document.

SID: A Segment Identifier which represents a specific segment in segment routing domain. The SID type used in this document is IPv6 address (also referenced as SRv6 Segment or SRv6 SID).

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:

- IPv6 header with source and destination addresses respectively SA and 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 S1 is the first SID and S3 is the last SID. (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 behavior, the (S3, S2, S1; SL) is more convenient.

- The payload of the packet is omitted.

SRH[SL] represents the SID pointed by the SL field in the first SRH. In our example, SRH[2] represents S1, SRH[1] represents S2 and SRH[0] represents S3.

FIB is the abbreviation for the forwarding table. A FIB lookup is a lookup in the forwarding table. When a packet is intercepted on a wire, it is possible that SRH[SL] is different from the DA.

3. 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 Local SID Table". This table contains all the local SRv6 segments explicitly instantiated at node N. N is the parent node for these SIDs.

A local SID of N can be an IPv6 address associated to a local interface of N but it is not mandatory. Nor is the My Local SID table populated by default with all IPv6 addresses defined on node N.

In most use-cases, a local SID will NOT be an address associated to a local interface of N.

A local SID of N could be routed to N but it does not have to be. Most often, it is routed to N via a shorter-mask prefix.

Let's provide a classic illustration.

Node N is configured with a loopback0 interface address of C1::1/40 originated in its IGP. Node N is configured with two SIDs: C1::100 and C2::101.

The entry C1::1 is not defined explicitly as an SRv6 SID and hence does not appear in the "My Local SID Table". The entries C1::100 and C2::101 are defined explicitly as SRv6 SIDs and hence appear in the "My Local SID Table".

The network learns about a path to C1::/40 via the IGP and hence a packet destined to C1::100 would be routed up to N. The network does not learn about a path to C2::/40 via the IGP and hence a packet destined to C2::101 would not be routed up to N.

A packet could be steered to a non-routed SID C2::101 by using a SID list <...,C1::100,C2::101,...> where the non-routed SID is preceded by a routed SID to the same node. This is similar to the local vs global segments in SR-MPLS.

Every SRv6 local SID instantiated has a specific instruction bound to it. This information is stored in the "My Local SID Table". The "My Local SID Table" has three main purposes:

- Define which local SIDs are explicitly instantiated

- Specify which instruction is bound to each of the instantiated SIDs

- Store the parameters associated with such instruction (i.e. OIF, NextHop,...)

We represent an SRv6 local 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 owns that SID.

Often, for simplicity of illustration, we will use a locator length of 64 bits. This is just an example. Implementations must not assume any a priori prefix length.

The FUNCT part of the SID is an opaque identification of a local function bound to the SID. Hence the name SRv6 Local SID.

A function may require additional arguments that would be placed in the rightmost-bits of the 128-bit space. In such case, the SRv6 Local SID will have the form LOC:FUNCT:ARGS.

These arguments may vary on a per-packet basis and may contain information related to the flow, service, or any other information required by the function associated to the SRv6 Local SID.

For to this reason, the "My Local SID Table" matches on a per longest-prefix-match basis.

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.

4. Functions associated with a Local SID

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

We define hereafter a set of well-known functions that can be associated with a SID.

End	Endpoint function The SRv6 instantiation of a prefix SID
End.X	Endpoint function with Layer-3 cross-connect The SRv6 instantiation of a Adj SID
End.T	Endpoint function with specific IPv6 table lookup
End.DX2	Endpoint with decapsulation and Layer-2 cross-connect L2VPN use-case
End.DX6	Endpoint with decapsulation and IPv6 cross-connect IPv6 L3VPN use (equivalent of a per-CE VPN label)
End.DX4	Endpoint with decapsulation and IPv4 cross-connect IPv4 L3VPN use (equivalent of a per-CE VPN label)
End.DT6	Endpoint with decapsulation and IPv6 table lookup IPv6 L3VPN use (equivalent of a per-VRF VPN label)
End.DT4	Endpoint with decapsulation and IPv4 table lookup IPv4 L3VPN use (equivalent of a per-VRF VPN label)
End.B6	Endpoint bound to an SRv6 policy SRv6 instantiation of a Binding SID
End.B6.Encaps	Endpoint bound to an SRv6 encapsulation Policy SRv6 instantiation of a Binding SID
End.BM	Endpoint bound to an SR-MPLS Policy SRv6/SR-MPLS instantiation of a Binding SID
End.S	Endpoint in search of a target in table T
End.AS	Endpoint to SR-unaware APP via static proxy
End.AM	Endpoint to SR-unaware APP via masquerading

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 function on the packet.

We call N the node who has an explicitly defined local SID S and we detail the function that N binds to S.

At the end of this section we also present some flavours of these well-known functions.

4.1. End: Endpoint

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

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

```
1.   IF NH=SRH and SL > 0
2.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      FIB lookup on updated DA                          ;; Ref1
5.      forward accordingly to the matched entry          ;; Ref2
6.   ELSE
7.      drop the packet                                   ;; Ref3
```

Ref1: The End function performs the FIB lookup in the forwarding table associated to the ingress interface

Ref2: If the FIB lookup matches a multicast state, then the related RPF check must be considered successful

Ref3: a local SID could be bound to a function which authorizes the decapsulation of an outer header (e.g. IPinIP) or the punting of the packet to TCP, UDP or any other protocol. This however needs to be explicitly defined in the function bound to the local SID. By default, a local SID bound to the well-known function "End" only allows the punting to OAM protocols and neither allows the decapsulation of an outer header nor the cleanup of an SRH. As a consequence, an End SID cannot be the last SID of an SRH and cannot be the DA of a packet without SRH.

This is the SRv6 instantiation of a Prefix SID [I-D.ietf-spring-segment-routing].

4.2. End.X: Endpoint with Layer-3 cross-connect

The "Endpoint with cross-connect to an array of layer-3 adjacencies" function (End.X for short) is a variant of the End function.

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

```
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   ;; Ref1
5.   ELSE
6.      drop the packet                                   ;; Ref2
```

Ref1: 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.

Ref2: An End.X function only allows punting to OAM and does not allow decaps. An End.X SID cannot be the last SID of an SRH and cannot be the DA of a packet without SRH.

The End.X function is required to express any traffic-engineering policy.

This is the SRv6 instantiation of an Adjacency SID [I-D.ietf-spring-segment-routing].

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 with SR-MPLS, an AdjSID is typically preceded by a PrefixSID. This is unlikely in SRv6 as most likely an End.X SID is globally routed to N.

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 for that bundle: one for the bundle itself and then up to one for each member link. This is the equivalent of the L2-Link Adj SID in SR-MPLS [I-D.ietf-isis-l2bundles].

4.3. End.T: Endpoint with specific IPv6 table lookup

The "Endpoint with specific IPv6 table lookup" function (End.T for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.T SID, N does:

```
1.   IF NH=SRH and SL > 0                                    ;; Ref1
2.      lookup the next segment in IPv6 table T associated with the SID
3.      forward via the matched table entry
4.   ELSE
5.      drop the packet
```

Ref1: The End.T SID must not be the last SID

The End.T is used for multi-table operation in the core.

4.4. End.DX2: Endpoint with decapsulation and Layer-2 cross-connect

The "Endpoint with decapsulation and Layer-2 cross-connect to OIF" function (End.DX2 for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DX2 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH  = 59                                     ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward the resulting frame via OIF associated to the SID
6.   ELSE
7.      drop the packet
```

Ref1: An End.DX2 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: We conveniently reuse the next-header value 59 allocated to IPv6 No Next Header [RFC2460]. When the SID is of function End.DX2 and the Next-Header=59, we know that an Ethernet frame is in the payload without any further header.

An End.DX2 function could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

The End.DX2 is used for L2VPN use-cases.

4.5. End.DX6: Endpoint with decapsulation and IPv6 cross-connect

The "Endpoint with decapsulation and cross-connect to an array of IPv6 adjacencies" function (End.DX6 for short) is a variant of the End and End.X functions.

When N receives a packet destined to S and S is a local End.DX6 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH = 41                                     ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward to layer-3 adjacency bound to the SID S    ;; Ref3
6.   ELSE
7.      drop the packet
```

Ref1: The End.DX6 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers

Ref3: Selected based on a hash of the packet's header (at least SA, DA, Flow Label)

The End.DX6 is used for L3VPN use-cases where a FIB lookup in a specific tenant table at the egress PE is not required. This would be equivalent to the per-CE VPN label in MPLS[RFC4364].

4.6. End.DX4: Endpoint with decapsulation and IPv4 cross-connect

The "Endpoint with decapsulation and cross-connect to an array of IPv4 adjacencies" function (End.DX4 for short) is a variant of the End and End.X functions.

When N receives a packet destined to S and S is a local End.DX4 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                   ;; Ref1
3.   ELSE IF ENH  =  4                                    ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      forward to layer-3 adjacency bound to the SID S   ;; Ref3
6.   ELSE
7.      drop the packet
```

Ref1: The End.DX6 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

Ref3: Selected based on a hash of the packet's header (at least SA, DA, Flow Label)

The End.DX4 is used for L3VPN use-cases where a FIB lookup in a specific tenant table at the egress PE is not required. This would be equivalent to the per-CE VPN label in MPLS[RFC4364].

4.7. End.DT6: Endpoint with decapsulation and specific IPv6 table lookup

The "Endpoint with decapsulation and specific IPv6 table lookup" function (End.DT6 for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.DT6 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH = 41                                      ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      lookup the exposed inner IPv6 DA in IPv6 table T
6.      forward via the matched table entry
7.   ELSE
8.      drop the packet
```

Ref1: the End.DT6 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 41 refers to IPv6 encapsulation as defined by IANA allocation for Internet Protocol Numbers

The End.DT6 is used for L3VPN use-cases where a FIB lookup in a specific tenant table at the egress PE is required. This would be 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 decaps (without VPN/tenant implication).

4.8. End.DT4: Endpoint with decapsulation and specific IPv4 table lookup

The "Endpoint with decapsulation and specific IPv4 table lookup" function (End.DT4 for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.DT4 SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      drop the packet                                    ;; Ref1
3.   ELSE IF ENH = 4                                       ;; Ref2
4.      pop the (outer) IPv6 header and its extension headers
5.      lookup the exposed inner IPv4 DA in IPv4 table T
6.      forward via the matched table entry
7.   ELSE
8.      drop the packet
```

Ref1: the End.DT4 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 refers to IPv4 encapsulation as defined by IANA allocation for Internet Protocol Numbers

The End.DT4 is used for L3VPN use-cases where a FIB lookup in a specific tenant table at the egress PE is required. This would be equivalent to the per-VRF VPN label in MPLS[RFC4364].

4.9. End.B6: Endpoint bound to an SRv6 policy

The "Endpoint bound to an SRv6 Policy" is a variant of the End function.

When N receives a packet destined to S and S is a local End.B6 SID, N does:

```
1.   IF NH=SRH and SL > 0                                  ;; Ref1
2.      do not decrement SL nor update the IPv6 DA with SRH[SL]
3.      insert a new SRH                                   ;; Ref2
4.      set the IPv6 DA to the first segment of the SRv6 Policy
5.      forward according to the first segment of the SRv6 Policy
6.   ELSE
7.      drop the packet
```

Ref1: An End.B6 SID, by definition, is never the last SID.

Ref2: In case that an SRH already exists, the new SRH is inserted in between the IPv6 header and the received SRH

Note: Instead of the term "insert", "push" may also be used.

The End.B6 function is required to express scalable traffic-engineering policies across multiple domains. This is the SRv6 instantiation of a Binding SID [I-D.ietf-spring-segment-routing].

4.10. End.B6.Encaps: Endpoint bound to an SRv6 encapsulation policy

This is a variation of the End.B6 behavior where the SRv6 Policy also includes an IPv6 Source Address A.

When N receives a packet destined to S and S is a local End.B6.Encaps SID, N does:

```
1.   IF NH=SRH and SL > 0
2.      decrement SL and update the IPv6 DA with SRH[SL]
3.      push an outer IPv6 header with its own SRH
4.      set the outer IPv6 SA to A
5.      set the outer IPv6 DA to the first segment of the SRv6 Policy
6.      forward according to the first segment of the SRv6 Policy
7.   ELSE
8.      drop the packet
```

Instead of simply inserting an SRH with the policy (End.B6), this behavior also adds an outer IPv6 header. The source address defined for the outer header does not have to be a local SID of the node.

4.11. End.BM: Endpoint bound to an SR-MPLS policy

The "Endpoint bound to an SR-MPLS Policy" is a variant of the End.B6 function.

When N receives a packet destined to S and S is a local End.BM SID, N does:

```
1.   IF NH=SRH and SL > 0                                    ;; Ref1
2.      decrement SL and update the IPv6 DA with SRH[SL]
3.      push an MPLS label stack <L1, L2, L3> on the received packet
4.      forward according to L1
5.   ELSE
6.      drop the packet
```

Ref1: an End.BM SID, by definition, is never the last SID.

The End.BM function 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/SR-MPLS instantiation of a Binding SID [I-D.ietf-spring-segment-routing].

4.12. End.S: Endpoint in search of a target in table T

The "Endpoint in search of a target in Table T" function (End.S for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.S SID, N does:

```
1.   IF NH=SRH and SL = 0                                    ;; Ref1
2.      drop the packet
3.   ELSE IF match(last SID) in specified table T
4.      forward accordingly
5.   ELSE
6.      apply the End behavior
```

Ref1: By definition, an End.S SID cannot be the last SID, as the last SID is the targeted object.

The End.S function is required in information-centric networking (ICN) use-cases where the last SID in the SRv6 SID list represents a targeted object. If the identification of the object would require more than 128 bits, then obvious customization of the End.S function may either use multiple SIDs or a TLV of the SR header to encode the searched object ID.

4.13. End.AS: Endpoint to SR-unaware APP via static proxy

The "Endpoint to SR-unaware App via Static PROXY" (End.AS for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.AS SID, it does:

```
1.   IF ENH=4 or ENH=41                                      ;; Ref1
2.      remove the (outer) IPv6 header and its extension headers
3.      forward via the interface associated with the LocalSID  ;; Ref2
4.   ELSE
5.      drop the packet
```

Ref1: 4 and 41 refer to IPv4 encapsulation and IPv6 encapsulation respectively as defined by IANA allocation for Internet Protocol Numbers

Ref2: An SR-unaware app resides in a VM, container or appliance behind this interface. We always assume that the packet that needs to be processed by the app is encapsulated in an outer IPv6 header with its SRH.

The End.AS supports service chaining through SR-unaware application.

When an End.AS SID is locally instantiated at N, it is assumed that the End.AS SID is associated with two interfaces, referred to as target and source interfaces, and an egress SRH. The target interface is the one described above. The source interface is used to recognize the packets coming back from the VM, container or appliance and is associated with an egress SRH. N encapsulates these packets in an outer IPv6 header with the configured egress SRH.

In this scenario, there are no restrictions on the operations that can be performed by the SR-unaware app on the stream of packets. The app can operate at all protocol layers (e.g. it can also terminate transport layer connections); the app can also generate new packets and initiate transport layer connections.

Note that is possible that the target and source interfaces coincide, (i.e. a single interface can be used to send and receive packets to/ from the VM, container or appliance). In this case, the VM, container or appliance can be inserted only in one "unidirectional" chain.

4.14. End.AM: Endpoint to SR-unaware APP via masquerading

The "Endpoint to SR-unaware App via Masquerading" (End.AM for short) is a variant of the End function.

When N receives a packet destined to S and S is a local End.AM SID, it does:

```
1.   IF NR=SRH & SL > 0                                        ;; Ref1
2.      decrement SL
3.      write the last SID in the IPv6 DA
4.      forward via the interface associated with the LocalSID  ;; Ref2
5.   ELSE
6.      drop the packet
```

Ref1: An End.AM must not be the last SID.

Ref2: An SR-unaware VNF resides behind this interface

The End.AM supports service chaining through SR-unaware application.

We "masquerade" the packet for two reasons:

1.- We want the app to receive a packet with the source and destination addresses respectively set as the true source and the final destination.

2.- We do not want the app to support/read the SRH. We leverage [RFC2460] which specifies that a transit node does not need to process an IPv6 routing extension header.

When an End.AM SID is locally instantiated at N, it is assumed that two interfaces are associated with the SID, referred to as target and source. The target interface is the one described above. The source interface identifies the packets coming back from the app. N is set to always inspect the SRH of the packets coming from the source interface and set their DA = SRH[SL] (to "de-masquerade" the SRH header).

In this scenario, we assume that the app residing in the VM, container or appliance can only inspect the packets, drop the packets, perform limited changes to the packet (in particular, the app must not change the IP Destination Address of the packet). The app cannot terminate a transport connection nor generate arbitrary packets. For example Firewalls, Intrusion Detection Systems, Deep Packet Inspectors are among the app classes that can be supported in this scenario.

4.15. SR-aware application

Generally, any SR-aware application can be bound to an SRv6 SID. This application could represent anything from a small piece of code focused on topological/tenant function to a much larger process focusing on higher-level applications (e.g. video compression, transcoding etc.).

The ways in which an SR-aware application can binds itself on a local SID depends on the operating system. Let us consider an SR-aware application running on a Linux operating system. A possible approach is to associate an SRv6 SID to a target (virtual) interface, so that packets with IP DA corresponding to the SID will be sent to the target interface. In this approach, the SR-aware application can simply listen to all packets received on the interface.

A different approach for the SR-aware app is to listen to packets received with a specific SRv6 SID as IPv6 DA on a given transport port (i.e. corresponding to a TCP or UDP socket). In this case, the app can read the SRH information with a getsockopt Linux system call and can set the SRH information to be added to the outgoing packets with a setsocksopt system call.

4.16. Flavours

We present the PSP and USP variants of the functions End, End.X and End.T. For each of these functions these variants can be enabled or disabled either individually or together.

4.16.1. PSP: penultimate segment POP of the SRH

After the instruction 'update the IPv6 DA with SRH[SL]' is executed, the following instructions must be added:

```
1.   IF updated SL = 0 & PSP is TRUE
2.      pop the top SRH                                        ;; Ref1
```

Ref1: The received SRH had SL=1. When the last SID is written in the DA, the End, End.X and End.T functions with the PSP flavour pop the first (top-most) SRH. Subsequent stacked SRH's may be present but are not processed as part of the function.

4.16.2. USP: Ultimate Segment Pop of the SRH

We insert at the beginning of the pseudo-code the following instructions:

```
1.   IF SL = 0 & NH=SRH & USP=TRUE                             ;; Ref1
2.      pop the top SRH
3.      restart the function processing on the modified packet  ;; Ref2
```

Ref1: The next header is an SRH header

Ref2: Typically SL of the exposed SRH is > 0 and hence the restarting of the complete function would lead to decrement SL, update the IPv6 DA with SRH[SL], FIB lookup on updated DA and forward accordingly to the matched entry.

5. Transit behaviors

We define hereafter the set of basic transit behaviors.

T	Transit behavior
T.Insert	Transit behavior with insertion of an SRv6 Policy
T.Encaps	Transit behavior with encapsulation in an SRv6 policy
T.Encaps.L2	T.Encaps behavior of the received L2 frame

This list can be expanded in case any new functionality requires it.

5.1. T: Transit behavior

As per [RFC2460], if a node N receives a packet (A, S2)(S3, S2, S1; SL=2) and S2 is neither a local address nor a local SID of N then N forwards the packet without inspecting the SRH.

This means that N treats the following two packets with the same performance:

- (A, S2)

- (A, S2)(S3, S2, S1; SL=2)

A transit node does not need to count by default the amount of transit traffic with an SRH extension header. This accounting might be enabled as an optional behavior leveraging SEC4 behavior described later in this document.Section 7.4

A transit node MUST include the outer flow label in its ECMP hash[RFC6437].

5.2. T.Insert: Transit with insertion of 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 SRv6 Policy with one SID list <S1, S2, S3>.

The "T.Insert" transit insertion behavior is defined as follows:

1. insert the SRH (B2, S3, S2, S1; SL=3) ;; Ref1, Ref1bis
2. set the IPv6 DA = S1
3. forward along the shortest path to S1

Ref1: The received IPv6 DA is placed as last SID of the inserted SRH.

Ref1bis: The SRH is inserted before any other IPv6 Routing Extension Header.

After the T.Insert behavior, P1 and P2 respectively look like:

- (A, S1) (B2, S3, S2, S1; SL=3)

- (A, S1) (B2, S3, S2, S1; SL=3) (B3, B2, B1; SL=1)

5.3. T.Encaps: Transit 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 Encapsulation Policy with a Source Address A and a Segment list <S1, S2, S3>.

The T.Encaps transit encapsulation behavior is defined as follows:

1. **push an IPv6 header with its own SRH (S3, S2, S1; SL=2)**
2. **set outer IPv6 SA = N and outer IPv6 DA = S1**
3. **set outer payload length, traffic class and flow label** ;; Ref 1
4. **update the next_header value** ;; Ref 1
5. **decrement inner Hop Limit or TTL** ;; Ref 1
6. **forward along the shortest path to S1**

After the T.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 T.Encaps behavior is valid for any kind of Layer-3 traffic. This behavior is commonly used for L3VPN with IPv4 and IPv6 deployements.

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.

Ref 1: As described in [RFC2473] (Generic Packet Tunneling in IPv6 Specification)

5.4. T.Encaps.L2: Transit with encapsulation of L2 frames

While T.Encaps encapsulates the received IP packet, T.Encaps.L2 encapsulates the received L2 frame (i.e. the received ethernet header and its optional VLAN header is in the payload of the outer packet).

If the outer header is pushed without SRH then the DA must be a SID of type End.DX2 and the next-header must be 59 (IPv6 NoNextHeader). The received Ethernet frame follows the IPv6 header and its extension headers.

Else, if the outer header is pushed with an SRH, then the last SID of the SRH must be of type End.DX2 and the next-header of the SRH must be 59 (IPv6 NoNextHeader). The received Ethernet frame follows the IPv6 header and its extension headers.

6. Operation

6.1. Counters

Any SRv6 capable node MUST implement the following set of combined counters (packets and bytes):

- Per entry of the "My Local SID Table":
  - Traffic that matched that SID and was processed correctly
  - Traffic that matched that SID and was NOT processed correctly
- Per SRv6 Policy:
  - Traffic steered into it and processed correctly
  - Traffic steered into it and NOT processed correctly

Furthermore, an SRv6 capable node maintains an aggregate counter tracking the IPv6 traffic that was received with a destination address matching a local interface address that is not a local SID and the next-header is SRH. We remind that this traffic is dropped as an interface address is not a local SID by default. A SID must be explicitly instantiated.

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 the destination address is updated and a FIB lookup is performed and multiple ECMP paths exist to the updated destination address.

This occurs when End.X is bound to an array of adjacencies.

This occurs when the packet is steered in an SR policy whose selected path has multiple SID lists [I-D.filsfils-spring-segment-routing-policy].

7. Basic security for intra-domain deployment

We use the following terminology:

An internal node is a node part of the domain of trust.

A border router is an internal node at the edge of the domain of trust.

An external interface is an interface of a border router towards another domain.

An internal interface is an interface entirely within the domain of trust.

The internal address space is the IP address block dedicated to internal interfaces.

An internal SID is a SID instantiated on an internal node.

The internal SID space is the IP address block dedicated to internal SIDs.

External traffic is traffic received from an external interface to the domain of trust.

Internal traffic is traffic the originates and ends within the domain of trust.

The purpose of this section is to document how a domain of trust can operate SRv6-based services for internal traffic while preventing any external traffic from accessing the internal SRv6-based services.

It is expected that future documents will detail enhanced security mechanisms for SRv6 (e.g. how to allow external traffic to leverage internal SRv6 services).

7.1. SEC 1

An SRv6 router MUST support an ACL on the external interface that drops any traffic with SA or DA in the internal SID space.

A provider would generally do this for its internal address space to prevent access to internal addresses and in order to prevent spoofing. The technique is extended to the local SID space.

The typical counters of an ACL are expected.

7.2. SEC 2

An SRv6 router MUST support an ACL with the following behavior:

```
1.   IF (DA == LocalSID) && (SA != internal address or SID space)
2.      drop
```

This prevents access to local SIDs from outside the operator's infrastructure. Note that this ACL may not be enabled in all cases. For example, specific SIDs can be used to provide resources to devices that are outside of the operator's infrastructure.

When an SID is in the form of LOC:FUNCT:ARGS the DA match should be implemented as a prefix match covering the argument space of the specific SID i.s.o. a host route.

The typical counters of an ACL are expected.

7.3. SEC 3

As per the End definition, an SRv6 router MUST only implement the End behavior on a local IPv6 address if that address has been explicitly enabled as a segment.

This address may or may not be associated with an interface. This address may or may not be routed. The only thing that matters is that the local SID must be explicitly instantiated and explicitly bound to a function (the default function is the End function).

7.4. SEC 4

An SRv6 router should support Unicast-RPF on source address on external interface.

This is a generic provider technique applied to the internal address space. It is extended to the internal SID space.

The typical counters to validate such filtering are expected.

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.

The specification of the SRv6 control-plane is outside the scope of this document.

We limit ourselves to a few important observations.

8.1. IGP

The End and End.X SIDs express topological functions and hence are expected to be signaled in the IGP together with the flavours PSP and USP.

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 classic, non-SID-related, IGP information. Routing is not governed neither influenced in any way by a SID advertisement in the IGP.

These two SIDs provide important topological functions for the IGP to build FRR/TI-LFA solution and for TE processes relying on IGP LSDB to build SR policies.

8.2. BGP-LS

BGP-LS is expected to be the key service discovery protocol. Every node is expected to advertise via BGP-LS its SRv6 capabilities (e.g. how many SIDs in can insert as part of an T.Insert behavior) and any locally instantiated SID[I-D.ietf-idr-bgp-ls-segment-routing-ext][I-D.ietf-idr-te-lsp-distribution].

8.3. BGP IP/VPN

The End.DX46, End.DT46 and End.DX2 SIDs are expected to be signaled in BGP.

8.4. Summary

The following table summarizes which SID would be signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN
End (PSP, USP)	X	X	
End.X (PSP, USP)	X	X	
End.T (PSP, USP)	X	X	
End.DX2		X	X
End.DX6		X	X
End.DX4		X	X
End.DT6		X	X
End.DT4		X	X
End.B6		X	
End.B6.Encaps		X	
End.B6.BM		X	
End.S		X	
End.AS		X	
End.AM		X	

Table 1: SRv6 LocalSID signaling

The following table summarizes which transit capability would be signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN
T		X	
T.Insert		X	
T.Encaps		X	
T.Encaps.L2		X	

Table 2: SRv6 transit behaviors signaling

The previous table describes generic capabilities. It does not describe specific instantiated SID.

For example, a BGP-LS advertisement of the T capability of node N would indicate that node N supports the basic transit behavior. The T.Insert behavior would describe the capability of node N to instantiation a T.Insert behavior, specifically it would describe how many SIDs could be inserted by N without significant performance degradation. Same for T.Encaps (the number potentially lower as the overhead of the additional outer IP header is accounted).

The reader should also remember that any specific instantiated SR policy (via T.Insert or T.Encaps) is always assigned a Binding SID. He should remember that BSIDs are advertised in BGP-LS as shown in Table 1. Hence, it is normal that Table 2 only focuses on the generic capabilities related to T.Insert and T.Encaps as Table 1 advertises the specific instantiated BSID properties.

9. Illustration

We introduce a simplified SID allocation technique to ease the reading of the text. We document the reference diagram. We then illustrate the network programming concept through different use-cases. These use-cases have been thought to allow straightforward combination between each other.

9.1. Simplified SID allocation

To simplify the illustration, we assume:

A::/4 is dedicated to the internal SRv6 SID space

B::/4 is dedicated to the internal address space

We assume a location expressed in 48 bits and a function expressed in 80 bits

Node k has a classic IPv6 loopback address Bk::/128 which is advertised in the IGP

Node k has Ak::/48 for its local SID space. Its SIDs will be explicitly allocated from that block

Node k advertises Ak::/48 in its IGP

Function 0:0:0:0:0 (function 0, for short) represents the End function

Function 0:0:0:0:C2 (function C2, for short) represents the End.X function towards neighbor 2

Function 0:0:0:0:E100 (function E100, for short) represents the End.T function in tenant table 100

Each node K has:

An explicit SID instantiation Ak::0/128 bound to an End function with additional support for PSP and USP

An explicit SID instantiation Ak::Cj/128 bound to an End.X function to neighbor J with additional support for PSP and USP

9.2. Reference diagram

Let us assume the following topology where all the links have IGP metric 10 except the link 23 which is 100.

Nodes A, 1 to 8 and B are considered within the network domain while nodes CE-A and CE-B are outside the domain.

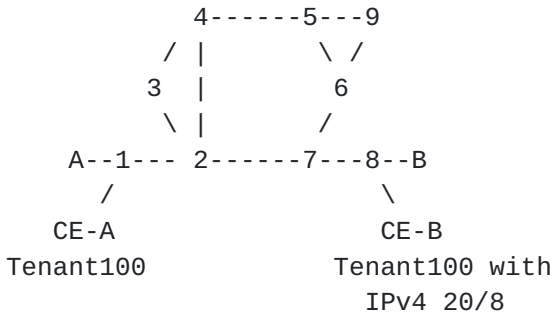

Figure 1: Reference topology

9.3. Basic security

Any edge node such as 1 would be configured with an ACL on any of its external interface (e.g. from CE-A) which drops any traffic with SA or DA in A::/4. See SEC 1 (Section 7.1).

Any core node such as 6 could be configured with an ACL with the SEC2 (Section 7.2) behavior "IF (DA == LocalSID) && (SA is not in A::/4 or B::/4) THEN drop".

SEC 3 (Section 7.3) protection is a default property of SRv6. A SID must be explicitly instantiated. In our illustration, the only available SIDs are those explicitly instantiated.

Any edge node such as 1 would be configured with Unicast-RPF on source address on external interface (e.g. from CE-A). See SEC 4 (Section 7.4).

9.4. SR-IPVPN

Let us illustrate the SR-IPVPN use-case applied to IPv4.

Nodes 1 and 8 are configured with a tenant 100, each respectively connected to CE-A and CE-B.

Node 8 is configured with a local SID A8::E100 of function End.DT4(100) bound to tenant IPv4 table 100.

Via BGP signaling or an SDN-based controller, Node 1's tenant-100 IPv4 table is programmed with an IPv4 SR-VPN route 20/8 via SRv6 policy <A8::E100>.

When 1 receives a packet P from CE-A destined to 20.20.20.20, P looks up its tenant-100 IPv4 table and finds an SR-VPN entry 20/8 via SRv6 policy <A8::E100>. As a consequence, 1 pushes an outer IPv6 header with SA=A1::0, DA=A8::E100 and NH=4. 1 then forwards the resulting packet on the shortest path to A8::/40.

When 8 receives the packet, 8 matches the DA in its My LocalSID table, finds the bound function End.DT4(100) and confirms NH=4. As a result, 8 decaps the outer header, looks up the inner IPv4 DA in tenant-100 IPv4 table, and forward the (inner) IPv4 packet towards CE-B.

The reader can easily infer all the other SR-IPVPN IP instantiations:

Route at ingress PE(1)	SR-VPN Egress SID of egress PE(8)
IPv4 tenant route with egress tenant table lookup	End.DT4 function bound to IPv4-tenant-100 table
IPv4 tenant route without egress tenant table lookup	End.DX4 function bound to CE-B (IPv4)
IPv6 tenant route with egress tenant table lookup	End.DT6 function bound to IPv6-tenant-100 table
IPv6 tenant route without egress tenant table lookup	End.DX6 function bound to CE-B (IPv6)

9.5. SR-Ethernet-VPWS

Let us illustrate the SR-Ethernet-VPWS use-case.

Node 1 is configured with an ethernet VPWS service:

- Local attachment circuit: Ethernet interface from CE-A

- Local End.DX2 bound to the local attachment circuit: A1::C2A

- Remote End.DX2 SID: A8::C2B

Node 8 is configured with an ethernet VPWS service:

- Local attachment circuit: Ethernet interface from CE-B

- Local End.DX2 bound to the local attachment circuit: A8::C2B

- Remote End.DX2 SID: A1::C2A

These configurations can either be programmed by an SDN controller or partially derived from a BGP-based signaling and discovery service .

When 1 receives a packet P from CE-A, 1 pushes an outer IPv6 header with SA=A1::0, DA=A8::C2B and NH=59. Note that no additional header is pushed. 1 then forwards the resulting packet on the shortest path to A8::/40.

When 8 receives the packet, 8 matches the DA in its My LocalSID table and finds the bound function End.DX2. After confirming that the next-header=59, 8 decaps the outer IPv6 header and forwards the inner Ethernet frame towards CE-B.

The reader can easily infer the Ethernet multipoint use-case:

Route at ingress PE(1)	SR-VPN Egress SID of egress PE(8)
Ethernet VPWS	End.DX2 function bound to CE-B (Ethernet)

9.6. SR TE for Underlay SLA

9.6.1. SR policy from the Ingress PE

Let's assume that node 1's tenant-100 IPv4 route "20/8 via A8::E100" is programmed with a color/community that requires low-latency underlay optimization [I-D.filsfils-spring-segment-routing-policy].

In such case, node 1 either computes the low-latency path to the egress node itself or delegates the computation to a PCE.

In either case, the location of the egress PE can easily be found by looking for who originates the SID block comprising the SID A8::E100. This can be found in the IGP's LSDB for a single domain case, and in the BGP-LS LSDB for a multi-domain case.

Let us assume that the TE metric encodes the per-link propagation latency. Let us assume that all the links have a TE metric of 10, except link 27 which has TE metric 100.

The low-latency path from 1 to 8 is thus 1245678.

This path is encoded in a SID list as: first a hop through A4::C5 and then a hop to 8.

As a consequence the SR-VPN entry 20/8 installed in the Node1's Tenant-100 IPv4 table is: T.Encaps with SRv6 Policy <A4::C5, A8::E100>.

When 1 receives a packet P from CE-A destined to 20.20.20.20, P looks up its tenant-100 IPv4 table and finds an SR-VPN entry 20/8. As a consequence, 1 pushes an outer header with SA=A1::0, A4::C5, NH=SRH followed by SRH (A8::E100, A4::C5; SL=1; NH=4) . 1 then forwards the resulting packet on the interface to 2.

2 forwards to 4 along the path to A4::/40.

When 4 receives the packet, 4 matches the DA in its My LocalSID table and finds the bound function End.X to neighbor 5. 4 notes the PSP capability of the SID A4::C5. 4 sets the DA to the next SID A8::E100. As 4 is the penultimate segment hop, it performs PSP and pops the SRH. 4 forwards the resulting packet to 5.

5, 6 and 7 forwards along the path to A8::/40.

When 8 receives the packet, 8 matches the DA in its My LocalSID Table and finds the bound function End.DT(100). As a result, 8 decaps the outer header, looks up the inner IPv4 DA in tenant-100 IPv4 table, and forward the (inner) IPv4 packet towards CE-B.

9.6.2. SR policy at a midpoint

Let us analyze a policy applied at a midpoint on a packet without SRH.

Packet P1 is (A1::, A8::E100).

Let us consider P1 when it is received by node 2 and let us assume that that node 2 is configured to steer A8::/40 in a transit behavior T.Insert associated with SR policy <A4::C5>.

In such a case, node 2 would send the following modified packet P1 on the link to 4:

(A1::, A4::C5)(A8::E100, A4::C5; SL=1).

The rest of the processing is similar to the previous section.

Let us analyze a policy applied at a midpoint on a packet with an SRH.

Packet P2 is (A1::, A7::)(A8::E100, A7::; SL=1).

Let us consider P2 when it is received by node 2 and let us assume that node 2 is configured to steer A7::/40 in a transit behavior T.Insert associated with SR policy <A4::C5, A9::>.

In such a case, node 2 would send the following modified packet P2 on the link to 4:

(A1::, A4::C5)(A7::, A9::, A4::C5; SL=2)(A8::E100, A7::; SL=1)

Node 4 would send the following packet to 5: (A1::, A9::)(A7::, A9::, A4::C5; SL=1)(A8::E100, A7::; SL=1)

Node 5 would send the following packet to 9: (A1::, A9::)(A7::, A9::, A4::C5; SL=1)(A8::E100, A7::; SL=1)

Node 9 would send the following packet to 6: (A1::, A7::)(A8::E100, A7::; SL=1)

Node 6 would send the following packet to 7: (A1::, A7::)(A8::E100, A7::; SL=1)

Node 7 would send the following packet to 8: (A1::, A8::E100)

9.7. End-to-End policy with intermediate BSID

Let us now describe a case where the ingress VPN edge node steers the packet destined to 20.20.20.20 towards the egress edge node connected to the tenant100 site with 20/8, but via an intermediate SR Policy represented by a single routable Binding SID. Let us illustrate this case with an intermediate policy which both encodes underlay optimization for low-latency and the service chaining via two SR-aware container-based apps.

Let us assume that the End.B6 SID A2::B1 is configured at node 2 and is associated with midpoint T.Insert policy <A4::C5, A9::A1, A6::A2>.

A4::C5 realizes the low-latency path from the ingress PE to the egress PE. This is the underlay optimization part of the intermediate policy.

A9::A1 and A6::A2 represent two SR-aware NFV applications residing in containers respectively connected to node 9 and 6.

Let us assume the following ingress VPN policy for 20/8 in tenant 100 IPv4 table of node 1: T.Encaps with SRv6 Policy <A2::B1, A8::E100>.

This ingress policy will steer the 20/8 tenant-100 traffic towards the correct egress PE and via the required intermediate policy that realizes the SLA and NFV requirements of this tenant customer.

Node 1 sends the following packet to 2: (A1::, A2::B1) (A8::E100, A2::B1; SL=1)

Node 2 sends the following packet to 4: (A1::, A4::C5) (A6::A2, A9::A1, A4::C5; SL=2)(A8::E100, A2::B1; SL=1)

Node 4 sends the following packet to 5: (A1::, A9::A1) (A6::A2, A9::A1, A4::C5; SL=1)(A8::E100, A2::B1; SL=1)

Node 5 sends the following packet to 9: (A1::, A9::A1) (A6::A2, A9::A1, A4::C5; SL=1)(A8::E100, A2::B1; SL=1)

Node 9 sends the following packet to 6: (A1::, A6::A2) (A8::E100, A2::B1; SL=1)

Node 6 sends the following packet to 7: (A1::, A8::E100)

Node 7 sends the following packet to 8: (A1::, A8::E100) which decaps and forwards to CE-B.

The benefits of using an intermediate Binding SID are well-known and key to the Segment Routing architecture: the ingress edge node needs to push fewer SIDs, the ingress edge node does not need to change its SR policy upon change of the core topology or re-homing of the container-based apps on different servers. Conversely, the core and service organizations do not need to share details on how they realize underlay SLA's or where they home their NFV apps.

9.8. TI-LFA

Let us assume two packets P1 and P2 received by node 2 exactly when the failure of link 27 is detected.

P1: (A1::, A7::)

P2: (A1::, A7::)(A8::E100, A7::; SL=1)

Node 2's pre-computed TI-LFA backup path for the destination A7:: is <A4::C5>. It is installed as a T.Insert transit behavior.

Node 2 protects the two packets P1 and P2 according to the pre-computed TI-LFA backup path and send the following modified packets on the link to 4:

P1: (A1::, A4::C5)(A7::, A4::C5; SL=1)

P2: (A1::, A4::C5)(A7::, A4::C5; SL=1) (A8::E100, A7::; SL=1)

Node 4 then sends the following modified packets to 5:

P1: (A1::, A7::)

P2: (A1::, A7::)(A8::E100, A7::; SL=1)

Then these packets follow the rest of their post-convergence path towards node 7 and then go to node 8 for the VPN decaps.

9.9. SR TE for Service chaining

We have illustrated the service chaining through SR-aware apps in a previous section.

We illustrate the use of End.AS functions to service chain an IP flow bound to the internet through two SR-unaware applications hosted in containers.

Let us assume that servers 20 and 70 are respectively connected to nodes 2 and 7. They are respectively configured with SID spaces A020::/40 and A070::/40. Their connected routers advertise the related prefixes in the IGP. Two SR-unaware container-based applications App2 and App7 are respectively hosted on server 20 and 70. Server 20 (70) is configured explicitly with an End.AS SID A020::2 for App2 (A070::7 for App7).

Let us assume a broadband customer with a home gateway CE-A connected to edge router 1. Router 1 is configured with an SR policy which encapsulates all the traffic received from CE-A into a T.Encaps policy <A020::2, A070::7, A8::E0> where A8::E0 is an End.DT4 SID instantiated at node 8.

P1 is a packet sent by the broadband customer to 1: (X, Y) where X and Y are two IPv4 addresses.

1 sends the following packet to 2: (A1::0, A020::2)(A8::E0, A070::7, A020::2; SL=2; NH=4)(X, Y).

2 forwards the packet to server 20.

20 recievs the packet (A1::0, A070::7)(A8::E0, A070::7, A020::2; SL=2; NH=1)(X, Y) and forwards the inner IPv4 packet (X,Y) to App2. App2 works on the packet and forwards it back to 20. 20 sends the (whole) IPv6 packet back to 2.

2 and 7 forward to server 70.

70 recieves the packet (A1::0, A8::E0)(X, Y) and forwards the inner IPv4 packet (X,Y) to App7. App7 works on the packet and forwards it back to 70. 70 sends the (whole) IPv6 packet back to 7.

7 forwards to 8.

8 performs the End.DT4 function and sends the IP packet (X, Y) towards its internet destination

10. Benefits

10.1. Seamless deployment

The VPN use-case can be realized with SRv6 capability deployed solely at the ingress and egress PE's.

All the nodes in between these PE's act as transit routers as per [RFC2460]. No software/hardware upgrade is required on all these nodes. They just need to support IPv6 per [RFC2460].

The SRTE/underlay-SLA use-case can be realized with SRv6 capability deployed at few strategic nodes.

It is well-known from the experience deploying SR-MPLS that underlay SLA optimization requires few SIDs placed at strategic locations. This was illustrated in our example with the low-latency optimization which required the operator to enable one single core node with SRv6 (node 4) where one single and End.X SID towards node 5 was instantiated. This single SID is sufficient to force the end-to-end traffic via the low-latency path.

The TI-LFA benefits are collected incrementally as SRv6 capabilities are deployed.

It is well-know that TI-LFA is an incremental node-by-node deployment. When a node N is enabled for TI-LFA, it computes TI-

LFA backup paths for each primary path to each IGP destination. In more than 50% of the case, the post-convergence path is loop-free and does not depend on the presence of any remote SRv6 SID. In the vast majority of cases, a single segment is enough to encode the post-convergence path in a loop-free manner. If the required segment is available (that node has been upgraded) then the related back-up path is installed in FIB, else the pre-existing situation (no backup) continues. Hence, as the SRv6 deployment progresses, the coverage incrementally increases. Eventually, when the core network is SRv6 capable, the TI-LFA coverage is complete.

The service chaining use-case can be realized with SRv6 capability deployed at few strategic nodes.

The service-chaining deployment is again incremental and does not require any pre-deployment of SRv6 in the network. When an NFV app A1 needs to be enabled for inclusion in an SRv6 service chain, all what is required is to install that app in a container or VM on an SRv6-capable server (Linux 4.10 or FD.io 17.04 release). The app can either be SR-aware or not, leveraging the proxy functions described in this document.

By leveraging the various END functions it can also be used to support any current PNF/VNF implementations and their forwarding methods (e.g. Layer 2).

The ability to leverage SR TE policies and BSIDs also permits building scalable, hierarchical service-chains.

10.2. Integration

The SRv6 network programming concept allows integrating all the application and service requirements: multi-domain underlay SLA optimization with scale, overlay VPN/Tenant, sub-50msec automated FRR, security and service chaining.

10.3. Security

The combination of well-known techniques (SEC1, SEC2, SEC4) and carefully chosen architectural rules (SEC3) ensure a secure deployment of SRv6 inside a multi-domain network managed by a single organization.

Inter-domain security will be described in a companion document.

11. IANA Considerations

This document has no actions for IANA.

12. Acknowledgements

TBD.

13. Contributors

Stefano Previdi, Dave Barach, Mark Townsley, Peter Psenak, Paul Wells, Robert Hanzl, Dan Ye, Patrice Brissette, Gaurav Dawra, Faisal Iqbal, Zafar Ali, 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 substantially contributed to the content of this document.

14. References

14.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

14.2. Informative References

[I-D.filsfils-spring-segment-routing-policy] Filsfils, C., Sivabalan, S., Yoyer, D., Nanduri, M., Lin, S., bogdanov@google.com, b., Horneffer, M., Clad, F., Steinberg, D., Decraene, B., and S. Litkowski, "Segment Routing Policy for Traffic Engineering", draft-filsfils-spring-segment-routing-policy-00 (work in progress), February 2017.

[I-D.ietf-6man-segment-routing-header] Previdi, S., Filsfils, C., Field, B., Leung, I., Linkova, J., Aries, E., Kosugi, T., Vyncke, E., and D. Lebrun, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-05 (work in progress), February 2017.

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Psenak, P., Filsfils, C., Gredler, H., Chen, M., and j. jefftant@gmail.com, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-01 (work in progress), February 2017.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Dong, J., Chen, M., Gredler, H., and j. jefftant@gmail.com, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-06 (work in progress), January 2017.

[I-D.ietf-isis-l2bundles]
Ginsberg, L., Bashandy, A., Filsfils, C., Previdi, S., Nanduri, M., and E. Aries, "Advertising L2 Bundle Member Link Attributes in IS-IS", draft-ietf-isis-l2bundles-03 (work in progress), February 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-11 (work in progress), February 2017.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, DOI 10.17487/RFC2460, December 1998, <http://www.rfc-editor.org/info/rfc2460>.

[RFC2473] Conta, A. and S. Deering, "Generic Packet Tunneling in IPv6 Specification", RFC 2473, DOI 10.17487/RFC2473, December 1998, <http://www.rfc-editor.org/info/rfc2473>.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://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, <http://www.rfc-editor.org/info/rfc6437>.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

John Leddy
Comcast
United States of America

Email: john_leddy@cable.comcast.com

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Dirk Steinberg
Steinberg Consulting
Germany

Email: dws@dirksteinberg.de

Robert Raszuk
Bloomberg LP
United States of America

Email: robert@raszuk.net

Satoru Matsushima
SoftBank Telecom
Japan

Email: satoru.matsushima@g.softbank.co.jp

David Lebrun
Universite catholique de Louvain
Belgium

Email: david.lebrun@uclouvain.be

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Netherlands

Email: bart.peirens@proximus.com

Stefano Salsano
Universita di Roma "Tor Vergata"
Italy

Email: stefano.salsano@uniroma2.it

Gaurav Naik
Drexel University
United States of America

Email: gn@drexel.edu

Hani Elmalky
Ericsson
United States of America

Email: hani.elmalky@ericsson.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

Arthi Ayyangar
Arista
United States of America

Email: arthi@arista.com

Satish Mynam
Dell Force10 Networks
United States of America

Email: satish_mynam@dell.com

Ahmed Bashandy
Cisco Systems, Inc.
United States of America

Email: bashandy@cisco.com

Kamran Raza
Cisco Systems, Inc.
Canada

Email: skraza@cisco.com

Darren Dukes
Cisco Systems, Inc.
Canada

Email: ddukes@cisco.com

Francois Clad
Cisco Systems, Inc.
France

Email: fclad@cisco.com

Pablo Camarillo Garvia (editor)
Cisco Systems, Inc.
Spain

Email: pcamaril@cisco.com