SPRING
Internet-Draft
Intended status: Standards Track
Expires: April 25, 2019

C. Filsfils
P. Camarillo, Ed.
Cisco Systems, Inc.
J. Leddy
Comcast
D. Voyer
Bell Canada
S. Matsushima
SoftBank
Z. Li
Huawei Technologies
October 22, 2018

SRv6 Network Programming
draft-filsfils-spring-srv6-network-programming-06

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 https://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 April 25, 2019.

Table of Contents

1. Introduction

Segment Routing leverages the source routing paradigm. An ingress node steers a packet through a 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. The network programming consists in combining segment routing functions, both simple and complex, to achieve a networking objective that goes beyond mere packet routing.

This document illustrates the SRv6 Network Programming concept and aims at standardizing the main segment routing functions to enable the creation of interoperable overlays with underlay optimization and service programming.

Familiarity with the Segment Routing Header [I-D.ietf-6man-segment-routing-header] is assumed.

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 are no other extension headers 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 address SA, destination addresses DA and SRH as next-header

- 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 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[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 SID Table". This table contains all the 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 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 A:1::/32 originated in its IGP. Node N is configured with two SIDs: B:1:100:: and B:2:101::.

The entry A:1:: is not defined explicitly as an SRv6 SID and hence does not appear in the "My SID Table". The entries B:1:100:: and B:2:101:: are defined explicitly as SRv6 SIDs and hence appear in the "My SID Table".

The network learns about a path to B:1::/32 via the IGP and hence a packet destined to B:1:100:: would be routed up to N. The network does not learn about a path to B:2::/32 via the IGP and hence a packet destined to B:2:101:: would not be routed up to N.

A packet could be steered to a non-routed SID B:2:101:: by using a SID list <...,B:1:100::,B:2:101::,...> 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 SID instantiated has a specific instruction bound to it. This information is stored in the "My SID Table". The "My SID Table" has three main purposes:

- Define which SIDs are explicitly instantiated on that node

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

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

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. The FUNCT value zero is invalid.

Often, for simplicity of illustration, we will use a locator length of 32 bits. This is just an example. Implementations must not assume any a priori prefix length.

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

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 SID.

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 SID

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

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 with Layer-3 cross-connect The SRv6 instantiation of a Adj SID
End.T	Endpoint with specific IPv6 table lookup
End.DX2	Endpoint with decaps and L2 cross-connect e.g. L2VPN use-case
End.DX2V	Endpoint with decaps and VLAN L2 table lookup EVPN Flexible cross-connect use-cases
End.DT2U	Endpoint with decaps and unicast MAC L2table lookup EVPN Bridging unicast use-cases
End.DT2M	Endpoint with decaps and L2 table flooding EVPN Bridging BUM use-cases with ESI filtering
End.DX6	Endpoint with decaps 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 decaps and IPv6 table lookup e.g. IPv6-L3VPN (equivalent to per-VRF VPN label)
End.DT4	Endpoint with decaps and IPv4 table lookup e.g. IPv4-L3VPN (equivalent to per-VRF VPN label)
End.DT46	Endpoint with decaps and IP table lookup e.g. IP-L3VPN (equivalent to per-VRF VPN label)
End.B6.Insert	Endpoint bound to an SRv6 policy SRv6 instantiation of a Binding SID
End.B6.Insert.RED	[...] with reduced SRH insertion SRv6 instantiation of a Binding SID
End.B6.Encaps	Endpoint bound to an SRv6 policy with encaps SRv6 instantiation of a Binding SID
End.B6.Encaps.RED	[...] with reduced SRH insertion SRv6 instantiation of a Binding SID
End.BM	Endpoint bound to an SR-MPLS Policy SRv6 instantiation of an SR-MPLS Binding SID
End.S	Endpoint in search of a target in table T

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 instantiated 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 the updated DA                    ;; Ref1
5.      forward accordingly to the matched entry        ;; Ref2
6.   ELSE
7.      drop the packet
```

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

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: 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
```

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.

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].

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.

4.3. End.T: 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.      decrement SL
3.      update the IPv6 DA with SRH[SL]
4.      lookup the next segment in IPv6 table T associated with the SID
5.      forward via the matched table entry
6.   ELSE
7.      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: Decapsulation and L2 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 to OIF bound to the SID S
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 [RFC8200]. When the SID corresponds to function End.DX2 and the Next-Header value is 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.

One of the applications of the End.DX2 function is the L2VPN/EVPN VPWS use-case.

4.5. End.DX2V: Decapsulation and VLAN L2 table lookup

The "Endpoint with decapsulation and specific VLAN table lookup" function (End.DX2V for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DX2V 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.      lookup the exposed inner VLANs in L2 table T
6.      forward via the matched table entry
7.   ELSE
8.      drop the packet
```

Ref1: An End.DX2V 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 [RFC8200]. When the SID corresponds to function End.DX2V and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

An End.DX2V function could be customized to expect a specific VLAN format and rewrite the egress VLAN header before forwarding on the outgoing interface.

The End.DX2V is used for EVPN Flexible cross-connect use-cases.

4.6. End.DT2U: Decapsulation and unicast MAC L2 table lookup

The "Endpoint with decapsulation and specific unicast MAC L2 table lookup" function (End.DT2U for short) is a variant of the endpoint function.

When N receives a packet destined to S and S is a local End.DT2U 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.      learn the exposed inner MAC SA in L2 table T          ;; Ref3
6.      lookup the exposed inner MAC DA in L2 table T
7.      IF matched entry in table T
8.         forward via the matched table T entry
9.      ELSE
10.        forward via all L2OIF entries in table T
11.  ELSE
12.     drop the packet
```

Ref1: An End.DT2U 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 [RFC8200]. When the SID corresponds to function End.DT2U and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

Ref3: In EVPN, the learning of the exposed inner MAC SA is done via control plane.

The End.DT2U is used for EVPN Bridging unicast use cases.

4.7. End.DT2M: Decapsulation and L2 table flooding

The "Endpoint with decapsulation and specific L2 table flooding" function (End.DT2M for short) is a variant of the endpoint function.

This function may take an argument: "Arg.FE2". It is an argument specific to EVPN ESI filtering. It is used to exclude a specific OIF (or set of OIFs) from L2 table T flooding.

When N receives a packet destined to S and S is a local End.DT2M 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
3.      learn the exposed inner MAC SA in L2 table T       ;; Ref3
4.      forward on all L2OIF excluding the one specified in Arg.FE2
5.   ELSE
6.      drop the packet
```

Ref1: An End.DT2M 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 [RFC8200]. When the SID corresponds to function End.DT2M and the Next-Header value is 59, we know that an Ethernet frame is in the payload without any further header.

Ref3: In EVPN, the learning of the exposed inner MAC SA is done via control plane

The End.DT2M is used for EVPN Bridging BUM use-case with ESI filtering capability.

4.8. End.DX6: 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.X function.

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)

One of the applications of the End.DX6 function is the L3VPNv6 use-case 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].

4.9. End.DX4: 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.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.DX4 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)

One of the applications of the End.DX4 function is the L3VPNv4 use-case 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].

4.10. End.DT6: 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

One of the applications of the End.DT6 function is the L3VPNv6 use-case 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).

4.11. End.DT4: 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

One of the applications of the End.DT4 is the L3VPNv4 use-case 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.DT4 may be defined for the main IPv4 table in which case and End.DT4 supports the equivalent of an IPv4inIPv6 decaps (without VPN/tenant implication).

4.12. End.DT46: Decapsulation and specific IP table lookup

The "Endpoint with decapsulation and specific IP table lookup" function (End.DT46 for short) is a variant of the End.DT4 and End.DT6 functions.

When N receives a packet destined to S and S is a local End.DT46 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 IF ENH = 41                                     ;; Ref2
 8.      pop the (outer) IPv6 header and its extension headers
 9.      lookup the exposed inner IPv6 DA in IPv6 table T
10.      forward via the matched table entry
11.   ELSE
12.      drop the packet
```

Ref1: the End.DT46 SID must always be the last SID, or it can be the Destination Address of an IPv6 packet with no SRH header.

Ref2: 4 and 41 refer to IPv4 and IPv6 encapsulation respectively as defined by IANA allocation for Internet Protocol Numbers

One of the applications of the End.DT46 is the L3VPN use-case where a FIB lookup in a specific IP 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.DT46 may be defined for the main IP table in which case and End.DT46 supports the equivalent of an IPinIPv6 decaps (without VPN/tenant implication).

4.13. End.B6.Insert: 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.Insert 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.Insert 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.Insert 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].

4.14. End.B6.Insert.Red: [...] with reduced SRH insertion

This is an optimization of the End.B6.Insert function.

End.B6.Insert.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the pushed SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

4.15. End.B6.Encaps: Endpoint bound to an SRv6 policy w/ encaps

This is a variation of the End.B6.Insert 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.      set outer payload length, trafic class and flow label   ;; Ref1,2
7.      update the Next-Header value                            ;; Ref1
8.      decrement inner Hop Limit or TTL                        ;; Ref1
9.      forward according to the first segment of the SRv6 Policy
10.  ELSE
11.     drop the packet
```

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

Ref 2: As described in [RFC6437] (IPv6 Flow Label Specification)

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.

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.

4.16. End.B6.Encaps.Red: [...] with reduced SRH insertion

This is an optimization of the End.B6.Encaps function.

End.B6.Encaps.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the outer SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

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.

4.17. 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 instantiation of an SR-MPLS Binding SID [I-D.ietf-spring-segment-routing].

4.18. 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.

4.19. 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 larger process focusing on higher-level applications (e.g. video compression, transcoding etc.).

The ways in which an SR-aware application 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.

4.20. Non SR-aware application

[I-D.xuclad-spring-sr-service-programming] defines a set of additional functions in order to enable non SR-aware applications to be associated with an SRv6 SID.

4.21. 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.

4.21.1. 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 & O-bit = 0              ;; Ref1
2.      pop the top SRH                                       ;; Ref2
```

Ref1: If the SRH.Flags.O-bit or SRH.Flags.A-bit is set, PSP of the SRH is disabled. Section 6.1 specifies the pseudocode needed to process the SRH.Flags.O-bit.

Ref2: 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.

4.21.2. USP: Ultimate Segment Pop of the SRH

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

```
1.   IF NH=SRH & SL = 0 & 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. These behaviors are not bound to a SID and they correspond to source SR nodes or transit nodes [I-D.ietf-6man-segment-routing-header].

T	Transit behavior
T.Insert	Transit behavior with insertion of an SRv6 policy
T.Insert.Red	Transit behavior with reduced insert of an SRv6 policy
T.Encaps	Transit behavior with encapsulation in an SRv6 policy
T.Encaps.Red	Transit behavior with reduced encaps in an SRv6 policy
T.Encaps.L2	T.Encaps applied to received L2 frames
T.Encaps.L2.Red	T.Encaps.Red applied to received L2 frames

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

5.1. T: Transit behavior

As per [RFC8200], 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.

A transit node MUST include the outer flow label in its ECMP load-balancing 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.Insert.Red: Transit with reduced insertion

The T.Insert.Red behavior is an optimization of the T.Insert behavior. It is defined as follows:

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

T.Insert.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the pushed SRH. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

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

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

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

5.4. 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 T 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 = T and outer IPv6 DA = S1
3. set outer payload length, traffic class and flow label ;; Ref1,2
4. update the Next-Header value ;; Ref1
5. decrement inner Hop Limit or TTL ;; Ref1
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)

Ref 2: As described in [RFC6437] (IPv6 Flow Label Specification)

5.5. T.Encaps.Red: Transit with reduced encapsulation

The T.Encaps.Red behavior is an optimization of the T.Encaps behavior. It is defined as follows:

1. push an IPv6 header with its own SRH (S3, S2; SL=2)
2. set outer IPv6 SA = T and outer IPv6 DA = S1
3. set outer payload length, traffic class and flow label ;; Ref1,2
4. update the Next-Header value ;; Ref1
5. decrement inner Hop Limit or TTL ;; Ref1
6. forward along the shortest path to S1

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

Ref 2: As described in [RFC6437] (IPv6 Flow Label Specification)

T.Encaps.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the SRH of the pushed IPv6 packet. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

After the T.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 SRH MAY be omitted when the SRv6 Policy only contains one segment and there is no need to use any flag, tag or TLV.

5.6. 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, End.DX2V, End.DT2U or End.DT2M 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, End.DX2V, End.DT2U or End.DT2M and the next-header of the SRH must be 59 (IPv6 NoNextHeader). The received Ethernet frame follows the IPv6 header and its extension headers.

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.

5.7. T.Encaps.L2.Red: Transit with reduced encaps of L2 frames

The T.Encaps.L2.Red behavior is an optimization of the T.Encaps.L2 behavior.

T.Encaps.L2.Red will reduce the size of the SRH by one segment by avoiding the insertion of the first SID in the SRH of the pushed IPv6 packet. In this way, the first segment is only introduced in the DA and the packet is forwarded according to it.

Note that SL value is initially pointing to a non-existing segment in the SRH.

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 entry of the "My SID Table", 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 maintains an aggregate counter CNT-3 tracking the IPv6 traffic that was received with a destination address matching a local interface address that is not a locally instantiated SID and the next-header is SRH with SL>0. 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, 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 [I-D.filsfils-spring-segment-routing-policy].

6.3. OAM

[I-D.ali-spring-srv6-oam] defines the OAM behavior for SRv6. This includes the definition of the SRH Flag 'O-bit', as well as additional OAM Endpoint functions.

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 that 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 locally instantiated 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.

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 an SRv6 SID.

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.

Packets received with destination address representing a local interface that has not been enabled as an SRv6 SID MUST be dropped.

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, End.T and End.X SIDs express topological functions and hence are expected to be signaled in the IGP together with the flavours PSP and USP [I-D.bashandy-isis-srv6-extensions].

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 three 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.dawra-idr-bgpls-srv6-ext].

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 are expected to be signaled in BGP [I-D.dawra-idr-srv6-vpn].

8.4. Summary

The following table summarizes which SIDs are signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN/EVPN
End (PSP, USP)	X	X	
End.X (PSP, USP)	X	X	
End.T (PSP, USP)	X	X	
End.DX2		X	X
End.DX2V		X	X
End.DT2U		X	X
End.DT2M		X	X
End.DX6		X	X
End.DX4		X	X
End.DT6		X	X
End.DT4		X	X
End.DT46		X	X
End.B6.Insert		X	
End.B6.Insert.Red		X	
End.B6.Encaps		X	
End.B6.Encaps.Red		X	
End.B6.BM		X	
End.S		X	

Table 1: SRv6 locally instanted SIDs signaling

The following table summarizes which transit capabilities are signaled in which signaling protocol.

	IGP	BGP-LS	BGP IP/VPN/EVPN
T		X	
T.Insert	X	X	
T.Insert.Red	X	X	
T.Encaps	X	X	
T.Encaps.Red	X	X	
T.Encaps.L2		X	
T.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 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 perform 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 is potentially lower as the overhead of the additional outer IP header is accounted).

The reader should also remember that any specific instantiated SR policy is always assigned a Binding SID. They 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::/16 is dedicated to the internal address space

B::/16 is dedicated to the internal SRv6 SID space

We assume a location expressed in 32 bits and a function expressed in 16 bits

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

Node k has B:k::/32 for its local SID space. Its SIDs will be explicitly allocated from that block

Node k advertises B:k::/32 in its IGP

Function 0:0:1:: (function 1, for short) represents the End function with PSP support

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

Each node k has:

An explicit SID instantiation B:k:1::/128 bound to an End function with additional support for PSP

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

9.2. Reference diagram

Let us assume the following topology where all the links have IGP metric 10 except the link 3-4 which is 100.

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

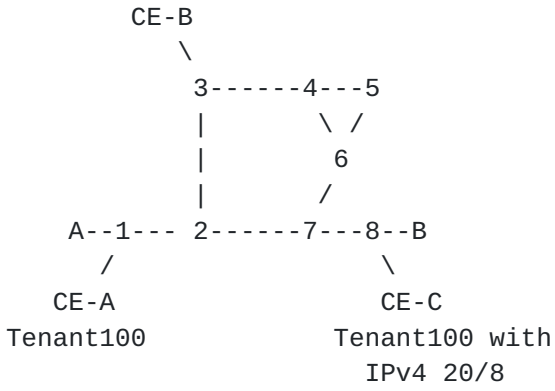


Figure 1: Reference topology

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 B::/16. See SEC-1 (Section 7.1).

Any core node such as 6 could be configured with an ACL with the SEC-2 (Section 7.2) behavior "IF (DA == LocalSID) && (SA is not in A::/16 or B::/16) 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.

9.4. SR-L3VPN

Let us illustrate the SR-L3VPN use-case applied to IPv4.

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

Node 8 is configured with a locally instantiated End.DT4 SID B:8:D100:: 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 <B:8:D100::>.

When 1 receives a packet P from CE-A destined to 20.20.20.20, 1 looks up 20.20.20.20 in its tenant-100 IPv4 table and finds an SR-VPN entry 20/8 via SRv6 policy <B:8:D100::>. As a consequence, 1 pushes an outer IPv6 header with SA=A:1::, DA=B:8:D100:: and NH=4. 1 then forwards the resulting packet on the shortest path to B:8::/32.

When 8 receives the packet, 8 matches the DA in its "My SID 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-C.

The reader can easily infer all the other SR-IPVPN 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-C (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-C (IPv6)

9.5. SR-Ethernet-VPWS

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

Node 8 is configured a locally instantiated End.DX2 SID B:8:DC2C:: bound to local attachment circuit {ethernet CE-C}.

Via BGP signalling or an SDN controller, node 1 is programmed with an Ethernet VPWS service for its local attachment circuit {ethernet CE-A} with remote endpoint B:8:DC2C::.

When 1 receives a frame F from CE-A, node 1 pushes an outer IPv6 header with SA=A:1::, DA=B:8:DC2C:: and NH=59. Note that no additional header is pushed. 1 then forwards the resulting packet on the shortest path to B:8::/32.

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

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

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

9.6. SR-EVPN-FXC

Let us illustrate the SR-EVPN-FXC use-case (Flexible cross-connect service).

Node 8 is configured with a locally instantiated End.DX2V SID B:8:DC2C:: bound to the L2 table T1. Node 8 is also configured with local attachment circuits {ethernet CE1-C VLAN:100} and {ethernet CE2-C VLAN:200} in table T1.

Via an SDN controller or derived from a BGP-based sginalling, the node 1 is programmed with an EVPN-FXC service for its local attachment circuit {ethernet CE-A} with remote endpoint B:8:DC2C::. For this purpose, the EVPN Type-1 route is used.

When node 1 receives a frame F from CE-A, it pushes an outer IPv6 header with SA=A:1::, DA=B:8:DC2C:: and NH=59. Note that no additional header is pushed. Node 1 then forwards the resulting packet on the shortest path to B:8::/32.

When node 8 receives the packet, it matches the IP DA in its "My SID Table" and finds the bound function End.DX2V. After confirming that next-header=59, node 8 decaps the outer IPv6 header, performs a VLAN loopkup in table T1 and forwards the inner Ethernet frame to matching interface e.g. for VLAN 100, packet is forwarded to CE1-C and for VLAN 200, frame is forwarded to CE2-C.

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

Route at ingress PE (1)	SR-VPN Egress SID of egress PE (8)
EVPN-FXC	End.DX2V function bound to CE1-C / CE2-C (Ethernet)

9.7. SR-EVPN

The following section details some of the particular use-cases of SR-EVPN. In particular bridging (unicast and multicast), multi-homing ESI filtering, L3 EVPN and EVPN-IRB.

9.7.1. EVPN Bridging

Let us illustrate the SR-EVPN unicast and multicast bridging.

Nodes 1, 3 and 8 are configured with a EVPN bridging service (E-LAN service).

Node 1 is configured with a locally instantiated End.DT2U SID B:1:D2AA:: bound to a local L2 table T1 where EVPN is enabled. This SID will be used to attract unicast traffic. Additionally, Node 1 is configured with a locally instantiated End.DT2M SID B:1:D2AF:: bound to the same local L2 table T1. This SID will be used to attract multicast traffic. Node 1 is also configured with local attachment circuit {ethernet CE-A VLAN:100} associated to table T1.

A similar instantiation is done at Node 4 and Node 8 resulting in:

- Node 1 - My SID table:

  - End.DT2U SID: B:1:D2AA:: table T1

  - End.DT2M SID: B:1:D2AF:: table T1

- Node 3 - My SID table:

  - End.DT2U SID: B:3:D2BA:: table T3

  - End.DT2M SID: B:3:D2BF:: table T3

- Node 8 - My SID table:

  - End.DT2U SID: B:8:D2CA:: table T8

  - End.DT2M SID: B:8:D2CF:: table T8

Nodes 1, 4 and 8 are going to exchange the End.DT2M SIDs via BGP-based EVPN Type-3 route. Upon reception of the EVPN Type-3 routes, each node build its own replication list per L2 table that will be used for ingress BUM traffic replication. The replication lists are the following:

- Node 1 - replication list: {B:3:D2BF:: and B:8:D2CF::}

- Node 3 - replication list: {B:1:D2AF:: and B:8:D2CF::}

- Node 8 - replication list: {B:1:D2AF:: and B:3:D2CF::}

When node 1 receives a BUM frame F from CE-A, it replicates that frame to every node in the replication list. For node 3, it pushes an outer IPv6 header with SA=A:1::, DA=B:3:D2BF:: and NH=59. For node 8, it performs the same operation but DA=B:8:D2CF::. Note that no additional headers are pushed. Node 1 then forwards the resulting packets on the shortest path for each destination.

When node 3 receives the packet, it matches the DA in its "My SID Table" and finds the bound function End.DT2M with its related layer2 table T3. After confirming that next-header=59, node 3 decaps the outer IPv6 header and forwards the inner Ethernet frame to all layer-2 output interface found in table T3. Similar processing is also performed by node 8 upon packet reception. This example is the same for any BUM stream coming from CE-B or CE-C.

Node 1,3 and 8 are also performing software MAC learning to exchange MAC reachability information (unicast traffic) via BGP among themselves.

Each MAC being learnt is exchanged using BGP-based EVPN Type-2 route.

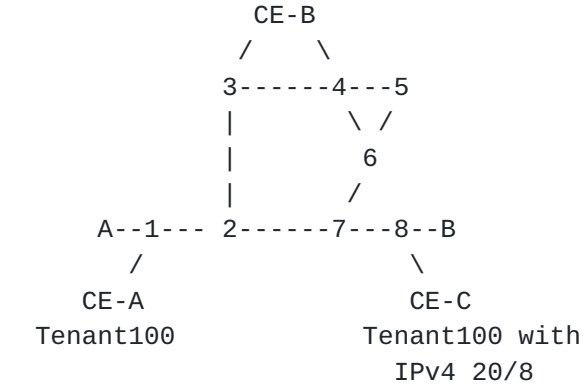
When node 1 receives an unicast frame F from CE-A, it learns its MAC-SA=CEA in software. Node 1 transmits that MAC and its associated SID B:1:D2AA:: using BGP-based EVPN route-type 2 to all remote nodes.

When node 3 receives an unicast frame F from CE-B destinated to MAC-DA=CEA, it performs a L2 lookup on T3 to find the associated SID. It pushes an outer IPv6 header with SA=A:3::, DA=B:1:D2AA:: and NH=59. Node 3 then forwards the resulting packet on the shortest path to B:1::/32. Similar processing is also performed by node 8.

9.7.2. EVPN Multi-homing with ESI filtering

In L2 network, support for traffic loop avoidance is mandatory. In EVPN all-active multi-homing scenario enforces that requirement using ESI filtering. Let us illustrate how it works:

Nodes 3 and 4 are peering partners of a redundancy group where the access CE-B, is connected in an all-active multi-homing way with these two nodes. Hence, the topology is the following:



EVPN ESI filtering - Reference topology

Nodes 3 and 4 are configured with an EVPN bridging service (E-LAN service).

Node 3 is configured with a locally instantiated End.DT2M SID B:3:D2BF:: bound to a local L2 table T1 where EVPN is enabled. This SID is also configured with the optional argument Arg.FE2 that specifies the attachment circuit. Particularly, node 3 assigns identifier 0xC1 to {ethernet CE-B}.

Node 4 is configured with a locally instantiated End.DT2M SID B:4:D2BF:: bound to a local L2 table T1 where EVPN is enabled. This SID is also configured with the optional argument Arg.FE2 that specifies the attachment circuit. Particularly, node 3 assigns identifier 0xC2 to {ethernet CE-B}.

Both End.DT2M SIDs are exchanged between nodes via BGP-based EVPN Type-3 routes. Upon reception of EVPN Type-3 routes, each node build its own replication list per L2 table T1.

On the other hand, the End.DT2M SID arguments (Arg.F2) are exchanged between nodes via SRv6 VPN SID attached to the BGP-based EVPN Type-1 route. The BGP ESI-filtering extended community label is set to implicit-null [I-D.dawra-idr-srv6-vpn].

Upon reception of EVPN Type-1 route and Type-3 route, node 3 merges merges the End.DT2M SID (B:4:D2BF:) with the Arg.FE2(0:0:0:C2::) from node 4 (its peering partner). This is done by a simple OR bitwise operation. As a result, the replication list on node 3 for the PEs 3,4 and 8 is: {B:1:D2AF::; B:4:D2BF:C2::; B:8:D2CF::}.

In a similar manner, the replication list on node 4 for the PEs 1,3 and 8 is: {B:1:D2AF::; B:3:D2BF:C1::; B:8:D2CF::}. Note that in this case the SID for PE3 contains the OR bitwise operation of SIDs B:3:D2BF:: and 0:0:0:C1::.

When node 3 receives a BUM frame F from CE-B, it replicates that frame to remote PEs. For node 4, it pushes an outer IPv6 header with SA=A:1::, DA=B:4:D2AF:C2:: and NH=59. Note that no additional header is pushed. Node 3 then forwards the resulting packet on the shortest path to node 4, and once the packet arrives to node 4, the End.DT2M function is executed forwarding to all L2 OIFs except the ones corresponding to identifier 0xC2.

9.7.3. EVPN Layer-3

EVPN layer-3 works exactly in the same way than L3VPN. Please refer to section Section 9.4

9.7.4. EVPN Integrated Routing Bridging (IRB)

EVPN IRB brings Layer-2 and Layer-3 together. It uses BGP-based EVPN Type-2 route to achieve Layer-2 intra-subnet and Layer-3 inter-subnet forwarding. The EVPN Type-2 route-2 maintains the MAC/IP association.

Node 8 is configured with a locally instantiated End.DT2U SID B:8:D2C:: used for unicast L2 traffic. Node 8 is also configured with locally instantiated End.DT4 SID B:8:D100:: bound to IPv4 tenant table 100.

Node 1 is going to be configured with the EVPN IRB service.

Node 8 signals to other remote PEs (1, 3) each ARP/ND request learned via BGP-based EVPN Type-2 route. For example, when node 8 receives an ARP/ND packet P from a host (20.20.20.20) on CE-C destined to 10.10.10.10, it learns its MAC-SA=CEC in software. It also learns the ARP/ND entry (IP SA=20.20.20.20) in its cache. Node 8 transmits that MAC/IP and its associated L3 SID (B:8:D100::) and L2 SID (B:8:D2C::).

When node 1 receives a packet P from CE-A destined to 20.20.20.20 from a host (10.10.10.10), node 1 looks up its tenant-100 IPv4 table and finds an SR-VPN entry for that prefix. As a consequence, node 1 pushes an outer IPv6 header with SA=A:1::, DA=B:8:D100:: and NH=4. Node 1 then forwards the resulting packet on the shortest path to B:8::/32. EVPN inter-subnet forwarding is then achieved.

When node 1 receives a packet P from CE-A destined to 20.20.20.20 from a host (10.10.10.11), P looks up its L2 table T1 MAC-DA lookup to find the associated SID. It pushes an outer IPv6 header with SA=A:1::, DA=B:8:D2C:: and NH=59. Note that no additional header is pushed. Node 8 then forwards the resulting packet on the shortest path to B:8::/32. EVPN intra-subnet forwarding is then achieved.

9.8. SR TE for Underlay SLA

9.8.1. SR policy from the Ingress PE

Let's assume that node 1's tenant-100 IPv4 route "20/8 via B:8:D100::" 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 locator comprising the SID B:8:D100::. 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 1234678.

This path is encoded in a SID list as: first a hop through B:3:C4:: 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 <B:3:C4::, B:8:D100::>.

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=A:1::, DA=B:3:C4::, NH=SRH followed by SRH (B:8:D100::, B:3:C4::; SL=1; NH=4). 1 then forwards the resulting packet on the interface to 2.

2 forwards to 3 along the path to B:3::/32.

When 3 receives the packet, 3 matches the DA in its "My SID Table" and finds the bound function End.X to neighbor 4. 3 notes the PSP capability of the SID B:3:C4::. 3 sets the DA to the next SID B:8:D100::. As 3 is the penultimate segment hop, it performs PSP and pops the SRH. 3 forwards the resulting packet to 4.

4, 6 and 7 forwards along the path to B:8::/32.

When 8 receives the packet, 8 matches the DA in its "My SID Table" and finds the bound function End.DT(100). As a result, 8 decaps the outer header, looks up the inner IPv4 DA (20.20.20.20) in tenant-100 IPv4 table, and forward the (inner) IPv4 packet towards CE-B.

9.8.2. SR policy at a midpoint

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

Packet P1 is (A:1::, B:8:D100::).

Let us consider P1 when it is received by node 2 and let us assume that that node 2 is configured to steer B:8::/32 in a T.Insert behavior associated with SR policy <B:3:C4::>.

In such a case, node 2 would send the following modified packet P1 on the link to 3:

(A:1::, B:3:C4::)(B:8:D100::, B:3:C4::; 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 (A:1::, B:7:1::)(B:8:D100::, B:7:1::; SL=1).

Let us consider P2 when it is received by node 2 and let us assume that node 2 is configured to steer B:7::/32 in a T.Insert behavior associated with SR policy <B:3:C4::, B:5:1::>.

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

(A:1::, B:3:C4::)(B:7:1::, B:5:1::, B:3:C4::; SL=2)(B:8:D100::, B:7:1::; SL=1)

Node 3 would send the following packet to 4: (A:1::, B:5:1::)(B:6:1::, B:5:1::, B:3:C4::; SL=1)(B:8:D100::, B:7:1::; SL=1)

Node 4 would send the following packet to 5: (A:1::, B:5:1::)(B:6:1::, B:5:1::, B:3:C4::; SL=1)(B:8:D100::, B:7:1::; SL=1)

Node 5 would send the following packet to 6: (A:1::, B:7:1::)(B:8:D100::, B:7:1::; SL=1)

Node 6 would send the following packet to 7: (A:1::, B:7:1::)(B:8:D100::, B:7:1::; SL=1)

Node 7 would send the following packet to 8: (A:1::, B:8:D100::)

9.9. 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 programming via two SR-aware container-based apps.

Let us assume that the End.B6.Insert SID B:2:B1:: is configured at node 2 and is associated with midpoint SR policy <B:3:C4::, B:9:A1::, B:6:A2::>.

B:3:C4:: realizes the low-latency path from the ingress PE to the egress PE. This is the underlay optimization part of the intermediate policy.

B:9:A1:: and B:6: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 <B:2:B1::, B:8:D100::>.

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: (A:1::, B:2:B1::) (B:8:D100::, B:2:B1::; SL=1)

Node 2 sends the following packet to 4: (A:1::, B:3:C4::) (B:6:A2::, B:9:A1::, B:3:C4::; SL=2)(B:8:D100::, B:2:B1::; SL=1)

Node 4 sends the following packet to 5: (A:1::, B:9:A1::) (B:6:A2::, B:9:A1::, B:3:C4::; SL=1)(B:8:D100::, B:2:B1::; SL=1)

Node 5 sends the following packet to 9: (A:1::, B:9:A1::) (B:6:A2::, B:9:A1::, B:3:C4::; SL=1)(B:8:D100::, B:2:B1::; SL=1)

Node 9 sends the following packet to 6: (A:1::, B:6:A2::) (B:8:D100::, B:2:B1::; SL=1)

Node 6 sends the following packet to 7: (A:1::, B:8:D100::)

Node 7 sends the following packet to 8: (A:1::, B:8:D100::) 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.

9.10. TI-LFA

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

P1: (A:1::, B:7:1::)

P2: (A:1::, B:7:1::)(B:8:D100::, B:7:1::; SL=1)

Node 2's pre-computed TI-LFA backup path for the destination B:7::/32 is <B:3:C4::>. 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: (A:1::, B:3:C4::)(B:7:1::, B:3:C4::; SL=1)

P2: (A:1::, B:3:C4::)(B:7:1::, B:3:C4::; SL=1) (B:8:D100::, B:7:1::; SL=1)

Node 4 then sends the following modified packets to 5:

P1: (A:1::, B:7:1::)

P2: (A:1::, B:7:1::)(B:8:D100::, B:7:1::; 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.

9.11. SR TE for Service programming

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

We illustrate the use of End.AS function [I-D.xuclad-spring-sr-service-programming] 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 B:20::/32 and B:70::/32. 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 A:20:2:: for App2 (A:70:7:: 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 <B:20:2::, B:70:7::, B:8:D0::> where B:8:D0:: 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::, B:20:2::)(B:8:D0::, B:70:7::, B:20:2::; SL=2; NH=4)(X, Y).

2 forwards the packet to server 20.

20 receives the packet (A1::, B:20:2::)(B:8:D0::, B:70:7::, B:20:2::; SL=2; NH=4)(X, Y) and forwards the inner IPv4 packet (X,Y) to App2. App2 works on the packet and forwards it back to 20. 20 pushes the outer IPv6 header with SRH (A1::, B:70:7::)(B:8:D0::, B:70:7::, B:20:2::; SL=1; NH=4) and sends the (whole) IPv6 packet with the encapsulated IPv4 packet back to 2.

2 and 7 forward to server 70.

70 receives the packet (A1::, B:70:7::)(B:8:D0::, B:70:7::, B:20:2::; SL=1; NH=4)(X, Y) and forwards the inner IPv4 packet (X,Y) to App7. App7 works on the packet and forwards it back to 70. 70 pushes the outer IPv6 header with SRH (A1::, B:8:D0::)(B:8:D0::, B:70:7::, B:20:2::; SL=0; NH=4) and sends the (whole) IPv6 packet with the encapsulated IPv4 packet back to 7.

7 forwards to 8.

8 receives (A1::, B:8:D0::)(B:8:D0::, B:70:7::, B:20:2::; SL=0; NH=4)(X, Y) and 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 [RFC8200]. No software/hardware upgrade is required on all these nodes. They just need to support IPv6 per [RFC8200].

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 programming use-case can be realized with SRv6 capability deployed at few strategic nodes.

The service-programming 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.

By leveraging the various End functions it can also be used to support any current VNF/CNF 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 programming.

10.3. Security

The combination of well-known techniques (SEC-1, SEC-2) and carefully chosen architectural rules (SEC-3) 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 requests the following new IANA registries:

- A new top-level registry "Segment-routing with IPv6 dataplane (SRv6) Parameters" to be created under IANA Protocol registries. This registry is being defined to serve as a top-level registry for keeping all other SRv6 sub-registries.

- A sub-registry "SRv6 Endpoint Behaviors" to be defined under top-level "Segment-routing with IPv6 dataplane (SRv6) 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 proceadure	Notes
0	0x0000	Reserved	Invalid
1-32767	0x0001-0x7FFF	IETF review	Draft Specifications
32768-49151	0x8000-0xBFFF	Reserved for experimental use	
49152-65534	0xC000-0xFFFE	Reserved for private use	
65535	0xFFFF	Reserved	Opaque

Table 3: SRv6 Endpoint Behaviors Registry

The initial registrations for the "Draft Specifications" portion of the sub-registry are as follows:

Value	Hex	Endpoint function	Reference
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	End.B6	[This.ID]
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	End.S	[This.ID]
26	0x001A	End.B6.Red	[This.ID]
27	0x001B	End.B6.Encaps.Red	[This.ID]

Table 4: IETF - SRv6 Endpoint Behaviors

12. Work in progress

We are working on a extension of this document to provide Yang modelling for all the functionality described in this document. This work is ongoing in [I-D.raza-spring-srv6-yang].

13. 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.

14. Contributors

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

Bruno Decraene
Orange
Frence

Email: bruno.decraene@orange.com

Bart Peirens
Proximus
Belgium

Email: bart.peirens@proximus.com

Hani Elmalky
Ericsson
United States of America

Email: hani.elmalky@gmail.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
Universite catholique de Louvain
Belgium

Email: david.lebrun@uclouvain.be

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
Arista
United States of America

Email: arthi@arista.com

Satish Mynam
Innovium Inc.
United States of America

Email: smynam@innovium.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

15. References

15.1. Normative References

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-14 (work in progress), June 2018.

[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>.

15.2. Informative References

[I-D.ali-spring-srv6-oam]
Ali, Z., Filsfils, C., Kumar, N., Pignataro, C., faiqbal@cisco.com, f., Gandhi, R., Leddy, J., Matsushima, S., Raszuk, R., daniel.voyer@bell.ca, d., Dawra, G., Peirens, B., Chen, M., and G. Naik, "Operations, Administration, and Maintenance (OAM) in Segment Routing Networks with IPv6 Data plane (SRv6)", draft-ali-spring-srv6-oam-01 (work in progress), July 2018.

[I-D.bashandy-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extensions to Support Routing over IPv6 Dataplane", draft-bashandy-isis-srv6-extensions-04 (work in progress), October 2018.

[I-D.dawra-idr-bgpls-srv6-ext]
Dawra, G., Filsfils, C., Talaulikar, K., Chen, M., daniel.bernier@bell.ca, d., Uttaro, J., Decraene, B., and H. Elmalky, "BGP Link State extensions for IPv6 Segment Routing(SRv6)", draft-dawra-idr-bgpls-srv6-ext-04 (work in progress), September 2018.

[I-D.dawra-idr-srv6-vpn]
Dawra, G., Filsfils, C., Dukes, D., Brissette, P., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., daniel.bernier@bell.ca, d., Steinberg, D., Raszuk, R., Decraene, B., Matsushima, S., and S. Zhuang, "BGP Signaling of IPv6-Segment-Routing-based VPN Networks", draft-dawra-idr-srv6-vpn-04 (work in progress), June 2018.

[I-D.filsfils-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., Hegde, S., daniel.voyer@bell.ca, d., Lin, S., bogdanov@google.com, b., Krol, P., Horneffer, M., Steinberg, D., Decraene, B., Litkowski, S., Mattes, P., Ali, Z., Talaulikar, K., Liste, J., Clad, F., and K. Raza, "Segment Routing Policy Architecture", draft-filsfils-spring-segment-routing-policy-06 (work in progress), May 2018.

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-08 (work in progress), May 2018.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Talaulikar, K., Dong, J., Chen, M., Gredler, H., and J. Tantsura, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-09 (work in progress), June 2018.

[I-D.ietf-isis-l2bundles]
Ginsberg, L., Bashandy, A., Filsfils, C., Nanduri, M., and E. Aries, "Advertising L2 Bundle Member Link Attributes in IS-IS", draft-ietf-isis-l2bundles-07 (work in progress), May 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[I-D.raza-spring-srv6-yang]
Raza, K., Rajamanickam, J., Liu, X., Hu, Z., Hussain, I., Shah, H., daniel.voyer@bell.ca, d., Elmalky, H., Matsushima, S., Horiba, K., and A. Abdelsalam, "YANG Data Model for SRv6 Base and Static", draft-raza-spring-srv6-yang-01 (work in progress), March 2018.

[I-D.xuclad-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., daniel.bernier@bell.ca, d., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", draft-xuclad-spring-sr-service-programming-00 (work in progress), July 2018.

[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>.

[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>.

[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>.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

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

Email: pcamaril@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

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