SPRING C. Filsfils
Internet-Draft P. Camarillo, Ed.
Intended status: Informational Cisco Systems, Inc.
Expires: February 15, 2020 Z. Li
Huawei Technologies
S. Matsushima
SoftBank
B. Decraene
Orange
D. Steinberg
Lapishills Consulting Limited
D. Lebrun
Google
R. Raszuk
Bloomberg LP
J. Leddy
Individual Contributor
August 14, 2019

Illustrations for SRv6 Network Programming
draft-filsfils-spring-srv6-net-pgm-illustration-01

Abstract

This document illustrates how SRv6 Network Programming [I-D.ietf-spring-srv6-network-programming] can be used to create interoperable and protected overlays with underlay optimization and service programming.

Requirements Language

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

Status of This Memo

This Internet-Draft will expire on February 15, 2020.

Copyright Notice

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.

[I-D.ietf-spring-srv6-network-programming] defines the SRv6 Network Programming concept and the main segment routing behaviors.

This document illustrates how these concepts can be used to enable the creation of interoperable overlays with underlay optimization and service programming.

The terminology for this document is defined in [I-D.ietf-spring-srv6-network-programming].

2. 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.

2.1. 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

2.2. 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.

```
        CE-B
           \
            3------4---5
            |       \ /
            |        6
            |       /
    A--1--- 2------7---8--B
      /                 \
   CE-A                 CE-C
Tenant100            Tenant100 with
                       IPv4 20/8
```

Figure 1: Reference topology

2.3. 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.

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

SEC-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.

2.4. 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)

2.5. 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)

2.6. 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)

2.7. 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.

2.7.1. 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.

2.7.2. 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:

```
                                  CE-B
                                 /    \
                                3------4---5
                                |       \ /
                                |        6
                                |       /
                        A--1--- 2------7---8--B
                          /                 \
                       CE-A                 CE-C
                    Tenant100            Tenant100 with
                                           IPv4 20/8
```

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.

2.7.3. EVPN Layer-3

EVPN layer-3 works exactly in the same way than L3VPN. Please refer to section Section 2.4

2.7.4. 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.

2.8. SR TE for Underlay SLA

2.8.1. 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.ietf-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.

2.8.2. 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::)

2.9. 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.

2.10. 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.

2.11. 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.

3. Benefits

3.1. 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.

3.2. 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.

3.3. 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.

4. 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.

5. Contributors

Daniel Bernier
Bell Canada
Canada

Email: daniel.bernier@bell.ca

Daniel Voyer
Bell Canada
Canada

Email: daniel.voyer@bell.ca

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

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

6. Informative References

[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 for SRv6 based Services.", draft-dawra-idr-srv6-vpn-05 (work in progress), October 2018.

[I-D.ietf-6man-segment-routing-header]
Filsfils, C., Dukes, D., Previdi, S., Leddy, J., Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment Routing Header (SRH)", draft-ietf-6man-segment-routing-header-22 (work in progress), August 2019.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Sivabalan, S., daniel.voyer@bell.ca, d., bogdanov@google.com, b., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-03 (work in progress), May 2019.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., daniel.voyer@bell.ca, d., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-01 (work in progress), July 2019.

[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-02 (work in progress), April 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <https://www.rfc-editor.org/info/rfc8200>.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Belgium

Email: cf@cisco.com

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

Email: pcamaril@cisco.com

Zhenbin Li
Huawei Technologies
China

Email: lizhenbin@huawei.com

Satoru Matsushima
SoftBank
1-9-1,Higashi-Shimbashi,Minato-Ku
Tokyo 105-7322
Japan

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

Bruno Decraene
Orange
France

Email: bruno.decraene@orange.com

Dirk Steinberg
Lapishills Consulting Limited
Cyprus

Email: dirk@lapishills.com

David Lebrun
Google
Belgium

Email: david.lebrun@uclouvain.be

Robert Raszuk
Bloomberg LP
United States of America

Email: robert@raszuk.net

John Leddy
Individual Contributor
United States of America

Email: john@leddy.net