Workgroup: Network Working Group
Internet-Draft:
draft-ietf-pim-sr-p2mp-policy-03
Published: 23 August 2021
Intended Status: Standards Track
Expires: 24 February 2022
Authors: D. Voyer, Ed. C. Filsfils
Bell Canada Cisco Systems, Inc.
R. Parekh H. Bidgoli Z. Zhang
Cisco Systems, Inc. Nokia Juniper Networks

Segment Routing Point-to-Multipoint Policy

Abstract

This document describes an architecture to construct a Point-to-Multipoint (P2MP) tree to deliver Multi-point services in a Segment Routing domain. A SR P2MP tree is constructed by stitching a set of Replication segments together. A SR Point-to-Multipoint (SR P2MP) Policy is used to define and instantiate a P2MP tree which is computed by a PCE.

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 24 February 2022.

Copyright Notice

Table of Contents

1. Introduction

A Multi-point service delivery could be realized via P2MP trees in a Segment Routing domain [RFC8402]. A P2MP tree spans from a Root node to a set of Leaf nodes via intermediate Replication Nodes. It consists of a Replication segment [I-D.ietf-spring-sr-replication-segment] at the root node, one or more Replication segments at Leaf nodes and intermediate Replication Nodes. The Replication segments are stitched together.

A Segment Routing P2MP policy, a variant of the SR Policy [I-D.ietf-spring-segment-routing-policy], is used to define a P2MP tree. A PCE is used to compute the tree from the Root node to the set of Leaf nodes via a set of Replication Nodes. The PCE then instantiates the P2MP tree in the SR domain by signaling Replication segments to Root, replication and Leaf nodes using various protocols (PCEP, BGP, NetConf etc.). Replication segments of a P2MP tree can be instantiated for SR-MPLS and SRv6 dataplanes.

2. P2MP Tree

A P2MP tree in a SR domain connects a Root to a set of Leaf nodes via a set of intermediate Replication Nodes. It consists of a Replication segment at the root stitched to Replication segments at intermediate Replication Nodes eventually reaching the Leaf nodes.

The Replication SID of the Replication segment at Root node is called Tree-SID. The Tree-SID SHOULD also be used as Replication SID of Replication segments at Replication and Leaf nodes. The Replication segments at Replication and Leaf nodes MAY use Replication SIDs that are not same as the Tree-SID.

The Replication segment at Root of a P2MP tree MUST be associated with that P2MP tree (i.e. <Root, Tree-ID> identifier in SR P2MP policy section below) to map a Multi-point service to the tree. A Replication segment that terminates a P2MP tree at a Leaf node MUST be associated with the P2MP tree to determine the context for a Multi-point service. The The information that can be used to derive this association is specific to encoding of the protocol (PCEP, BGP, NetConf etc.) used to instantiate the Replication segment for a P2MP tree. Replication segments at intermediate Replication Nodes of a tree are also associated with that tree.

For SR-MPLS, a PCE MAY decide not to instantiate Replication segments at Leaf nodes of a P2MP tree if it is known a priori that Multi-point services mapped to the P2MP tree can be identified using a context that is globally unique in SR domain. In this case, Replication Nodes connecting to Leaf nodes effectively does Penultimate-Hop Pop (PHP) behavior to pop Tree-SID from a packet. A

Multi-point service context assigned from "Domain-wide Common Block" (DCB) [I-D.ietf-bess-mvpn-evpn-aggregation-label] is an example of globally unique context.

A packet steered into a P2MP tree is replicated by the Replication segment at Root node to each downstream node in the Replication segment, with the Replication SID of the Replication segment at the downstream node. A downstream node could be a Leaf node or an intermediate Replication Node. In the latter case, replication continues with the Replication segments until all Leaf nodes are reached. A packet is steered into a P2MP tree in two ways:

*Based on a local policy-based routing at the Root node.

*Based on steering via the Tree-SID at the Root node.

2.1. Sharing Replication segments across P2MP trees

Two or more P2MP trees MAY share a Replication segment at Root or Replication Nodes if at minimum the first condition below is satisfied. A tree always has its own Replication segment at its root even if shares another Replication segment. A tree that shares another Replication segment may or may not have its own Replication segment on its Leaf nodes. If not, the second and third conditions apply to such situations.

1. The Leaf nodes reached via a shared Replication segment must be subset of Leaf or Replication Nodes of the P2MP trees that share this segment. Note if a Replication segment is shared, all its downstream Replication segments are also shared.

2. Some Multi-point services realized by the P2MP trees may need service context (e.g. packets are for certain VPNs, and/or from certain nodes). If the trees do not have their own Replication segments at their Leaf nodes then the packets transported on the P2MP trees MUST carry a service context that does not rely on the tree or root identification, e.g. a service label assigned from Domain-wide Common Block or common SRGB for SR-MPLS.

3. For some Multi-point services using P2MP trees that share Replication segments, packets transported on these trees MAY require a Tree context (e.g. MVPN Extranet [RFC7900] to avoid certain ambiguities - see Section 2.3.1 of RFC 7900). In this case, the trees MUST have their own Replication segments on the Leaf nodes. For SR-MPLS, this is similar to "tunnel stacking" concept.

Sharing of a Replication segment for P2MP trees is OPTIONAL. Exact procedures to ensure validity of above conditions across PM2P

services on nodes of a Segment Routing domain are outside the scope of this document.

3. SR P2MP Policy

The SR P2MP policy is a variant of an SR policy[I-D.ietf-spring-segment-routing-policy] and is used to instantiate SR P2MP trees.

A SR P2MP Policy is identified by the tuple <Root, Tree-ID>, where:

- *Root: The address of Root node of P2MP tree instantiated by the SR P2MP Policy

- *Tree-ID: A identifier that is unique in context of the Root. This is an unsigned 32-bit number.

A SR P2MP Policy is defined by following elements:

- *Leaf nodes: A set of nodes that terminate the P2MP trees.

- *Candidate Paths: See below.

A SR P2MP policy is provisioned on a PCE to instantiate the P2MP tree. The Tree-SID SHOULD be used as Binding SID of the P2MP policy. A PCE computes the P2MP tree and instantiates Replication segments at Root, Replication and Leaf nodes. When Replication segments are not shared across P2MP trees, the Root and Tree-ID of the SR P2MP policy are mapped to Replication-ID element of the Replication segment identifier i.e the SR Replication segment identifier is <Root, Tree-ID, Node-ID>. A shared Replication segment MAY be identified with zero Root-ID address (0.0.0.0 for IPv4 and :: for IPv6) and a Replication-ID that is unique in context of Node address where the Replication segment is instantiated when it is not associated a particular tree.

A SR P2MP Policy has one or more Candidate paths. The active Candidate path is selected based on the tie breaking rules amongst the candidate-paths as specified in[I-D.ietf-spring-segment-routing-policy]. Each candidate path has a set of topological/resource constraints and/or optimization objectives which determine the P2MP tree for that Candidate path. Tree-SID is an identifier of the P2MP tree of the candidate path in the forwarding plane. It is instantiated in the forwarding plane at Root node, intermediate Replication Nodes and Leaf nodes. The Tree-SID MAY be different at Replication and Leaf nodes.

4. Using Controller to build a P2MP Tree

A P2MP tree can be built using a Path Computation Element (PCE). This section outlines a high-level architecture for such an approach.

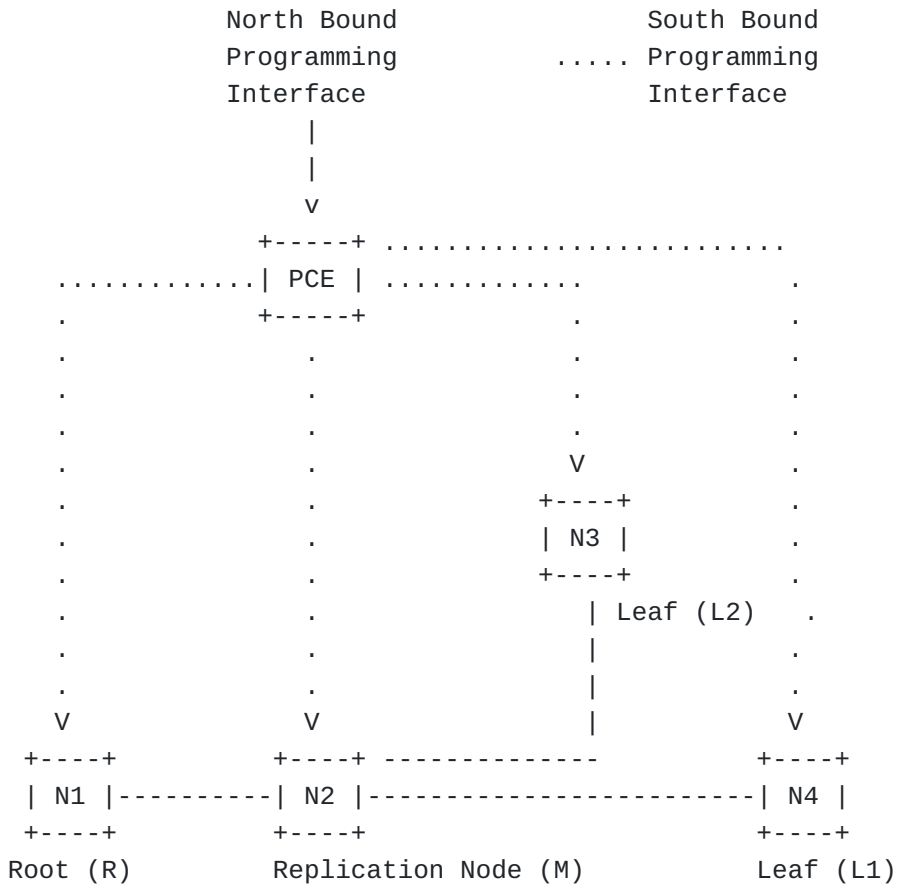

Figure 1: Centralized Control Plane Model

4.1. Provisioning SR P2MP Policy Creation

A SR P2MP policy can be instantiated and maintained in a centralized fashion using a Path Computation Element (PCE).

4.1.1. API

North-bound APIs on a PCE can be used to:

1. Create SR P2MP policy: CreateSRP2MPPolicy<Root, Tree-ID>

2. Delete SR P2MP policy: DeleteSRP2MPPolicy<Root, Tree-ID>

3. Modify SR P2MP policy Leaf Set: SRP2MPPolicyLeafSetModify<Root, Tree-ID, {Leaf Set}>

4. Create a Candidate Path for SR P2MP policy: CreateSRP2MPCandidatePath<Root, Tree-ID, <CP-ID>>

5. Delete a Candidate Path for SR P2MP policy: DeleteSRP2MPCandidatePath<Root, Tree-ID, <CP-ID>>

6. Update a Candidate Path for SR P2MP policy: UpdateSRP2MPCandidatePath<Root, Tree-ID, <CP-ID>, Preference, [Constraints], [Optimization], ...>

CP-ID is identifier of a Candidate Path within a SR P2MP policy. One possible identifier is the tuple <Protocol-Origin, originator, discriminator> as specified in [I-D.ietf-spring-segment-routing-policy].

Note these are conceptual APIs. Actual implementations may offer different APIs as long as they provide same functionality. For example, API might allow symbolic name to be assigned for a P2MP policy or APIs might allow individual Leaf nodes to be added or deleted from a policy instead of an update operation.

4.1.2. Invoking API

Interaction with a PCE can be via PCEP, REST, Netconf, gRPC, CLI. Yang model shall be be developed for this purpose as well.

4.2. P2MP Tree Computation

An entity (an operator, a network node or a machine) provisions a SR P2MP policy by specifying the addresses of the root (R) and set of leaves {L} as well as Traffic Engineering (TE) attributes of Candidate paths via a suitable North-Bound API. The PCE computes the tree of Active candidate path. The PCE MAY compute P2MP trees for all Candidate paths., If tree computation is successful, PCE instantiates the P2MP tree(s) using Replication segments on Root, Replication, and Leaf nodes.

Candidate path constraints shall include link color affinity, bandwidth, disjointness (link, node, SRLG), delay bound, link loss, etc. Candidate path shall be optimized based on IGP or TE metric or link latency.

The Tree SID of Candidate path of a SR P2MP policy can be either dynamically allocated by the PCE or statically assigned by entity provisioning the SR P2MP policy. Ideally, same Tree-SID SHOULD be used for Replication segments at Root, Replication, and Leaf nodes.

Different Tree-SIDs MAY be used at Replication Node(s) if it is not feasible to use same Tree SID.

A PCE can modify a P2MP tree following network element failure or in case a better path can be found based on the new network state. In this case, the PCE may want to setup the new instance of the tree and remove the old instance of the tree from the network in order to minimize traffic loss. In this case, the instances of trees for all the Candidate paths of a P2MP policy can be identified by an Instance-ID which is unique in context of the P2MP policy. As such, the identifier of non-shared Replication segments used to instantiate these trees becomes <Root-ID, Tree-ID, Node-ID, Instance-ID>.

A PCE shall be capable of computing paths across multiple IGP areas or levels as well as Autonomous Systems (ASs).

4.2.1. Topology Discovery

A PCE shall learn network topology, TE attributes of link/node as well as SIDs via dynamic routing protocols (IGP and/or BGP-LS). It may be possible for entities to pass topology information to PCE via north-bound API.

4.2.2. Capability and Attribute Discovery

It shall be possible for a node to advertise SR P2MP tree capability via IGP and/or BGP-LS. Similarly, a PCE can also advertise its P2MP tree computation capability via IGP and/or BGP-LS. Capability advertisement allows a network node to dynamically choose one or more PCE(s) to obtain services pertaining to SR P2MP policies, as well a PCE to dynamically identify SR P2MP tree capable nodes.

4.3. Instantiating P2MP tree on nodes

Once a PCE computes a P2MP tree for Candidate path of SR P2MP policy, it needs to instantiate the tree on the relevant network nodes via Replication segments. The PCE can use various protocols to program the Replication segments as described below.

4.3.1. PCEP

PCE Protocol (PCEP)has been traditionally used:

1. For a head-end to obtain paths from a PCE.

2. A PCE to instantiate SR policies.

PCEP protocol can be stateful in that a PCE can have a stateful control of an SR policy on a head-end which has delegated the

control of the SR policy to the PCE. PCEP shall be extended to provision and maintain SR P2MP trees in a stateful fashion.

4.3.2. BGP

BGP has been extended to instantiate and report SR policies. It shall be extended to instantiate and maintain P2MP trees for SR P2MP policies.

4.3.3. NetConf

TBD

4.4. Protection

4.4.1. Local Protection

A network link, node or path on the tree of a P2MP tree can be protected using SR policies computed by PCE. The backup SR policies shall be programmed in forwarding plane in order to minimize traffic loss when the protected link/node fails. It is also possible to use node local Fast Re-Route protection mechanisms (LFA) to protect link/nodes of P2MP tree.

4.4.2. Path Protection

It is possible for PCE create a disjoint backup tree for providing end-to-end path protection.

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

There are no additional security risks introduced by this design.

7. Acknowledgements

The authors would like to acknowledge Siva Sivabalan, Mike Koldychev and Vishnu Pavan Beeram for their valuable inputs..

8. Contributors

Clayton Hassen Bell Canada Vancouver Canada

Email: clayton.hassen@bell.ca

Kurtis Gillis Bell Canada Halifax Canada

Email: kurtis.gillis@bell.ca

Arvind Venkateswaran Cisco Systems, Inc. San Jose US

Email: arvvenka@cisco.com

Zafar Ali Cisco Systems, Inc. US

Email: zali@cisco.com

Swadesh Agrawal Cisco Systems, Inc. San Jose US

Email: swaagraw@cisco.com

Jayant Kotalwar Nokia Mountain View US

Email: jayant.kotalwar@nokia.com

Tanmoy Kundu Nokia Mountain View US

Email: tanmoy.kundu@nokia.com

Andrew Stone Nokia Ottawa Canada

Email: andrew.stone@nokia.com

Tarek Saad Juniper Networks Canada

Email:tsaad@juniper.net

9. References

9.1. Normative References

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", Work in Progress, Internet-Draft, draft-ietf-spring-segment-routing-policy-13, 28 May 2021, <https://www.ietf.org/archive/id/draft-ietf-spring-segment-routing-policy-13.txt>.

[I-D.ietf-spring-sr-replication-segment]
(editor), D. V., Filsfils, C., Parekh, R., Bidgoli, H., and Z. Zhang, "SR Replication Segment for Multi-point Service Delivery", Work in Progress, Internet-Draft, draft-ietf-spring-sr-replication-segment-05, 20 August 2021, <https://www.ietf.org/archive/id/draft-ietf-spring-sr-replication-segment-05.txt>.

[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>.

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/info/rfc8402>.

9.2. Informative References

[I-D.filsfils-spring-srv6-net-pgm-illustration] Filsfils, C., Garvia, P. C., Li, Z., Matsushima, S., Decraene, B., Steinberg, D., Lebrun, D., Raszuk, R., and J. Leddy, "Illustrations for SRv6 Network Programming", Work in Progress, Internet-Draft, draft-filsfils-spring-srv6-net-pgm-illustration-04, 30 March 2021, <https://www.ietf.org/archive/id/draft-filsfils-spring-srv6-net-pgm-illustration-04.txt>.

[I-D.ietf-bess-mvpn-evpn-aggregation-label] Zhang, Z., Rosen, E., Lin, W., Li, Z., and I. Wijnands, "MVPN/EVPN Tunnel Aggregation with Common Labels", Work in Progress, Internet-Draft, draft-ietf-bess-mvpn-evpn-aggregation-label-06, 19 April 2021, <https://www.ietf.org/archive/id/draft-ietf-bess-mvpn-evpn-aggregation-label-06.txt>.

[RFC7900] Rekhter, Y., Ed., Rosen, E., Ed., Aggarwal, R., Cai, Y., and T. Morin, "Extranet Multicast in BGP/IP MPLS VPNs", RFC 7900, DOI 10.17487/RFC7900, June 2016, <https://www.rfc-editor.org/info/rfc7900>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

Appendix A. Illustration of SR P2MP Policy and P2MP Tree

Consider the following topology:

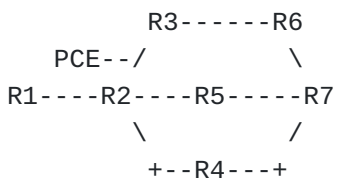

Figure 2: Figure 1

In these examples, the Node-SID of a node Rn is N-SIDn and Adjacency-SID from node Rm to node Rn is A-SIDmn. Interface between Rm and Rn is Lmn.

For SRv6, the reader is expected to be familiar with SRv6 Network Programming [RFC8986] to follow the examples. We use SID allocation scheme, reproduced below, from Illustrations for SRv6 Network Programming [I-D.filsfils-spring-srv6-net-pgm-illustration]

*2001:db8::/32 is an IPv6 block allocated by a RIR to the operator

*2001:db8:0::/48 is dedicated to the internal address space

*2001:db8:cccc::/48 is dedicated to the internal SRv6 SID space

*We assume a location expressed in 64 bits and a function expressed in 16 bits

*Node k has a classic IPv6 loopback address 2001:db8::k/128 which is advertised in the IGP

*Node k has 2001:db8:cccc:k::/64 for its local SID space. Its SIDs will be explicitly assigned from that block

*Node k advertises 2001:db8:cccc:k::/64 in its IGP

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

*Function :Cn:: (function Cn, for short) represents the End.X function to Node n

*Function :C1n: (function C1n for short) represents the End.X function to Node n with USD

Each node k has:

*An explicit SID instantiation 2001:db8:cccc:k:1::/128 bound to an End function with additional support for PSP

*An explicit SID instantiation 2001:db8:cccc:k:Cj::/128 bound to an End.X function to neighbor J with additional support for PSP

*An explicit SID instantiation 2001:db8:cccc:k:C1j::/128 bound to an End.X function to neighbor J with additional support for USD

Assume PCE is provisioned following SR P2MP policy at Root R1 with Tree-ID T-ID:

```
SR P2MP Policy <R1,T-ID>:
 Leaf Nodes: {R2, R6, R7}
 Candidate-path 1:
   Optimize: IGP metric
   Tree-SID: T-SID1
```

The PCE is responsible for P2MP tree computation. Assume PCE instantiates P2MP trees by signalling non-shared Replication segments i.e. Replication-ID of these Replication segments is <Root, Tree-ID>. If a Candidate-path can have multiple instances of P2MP trees, the Replication-ID is <Root, Tree-ID, Instance-ID>. In this example, we assume one instance of P2MP tree for a candidate-path. All Replication segments use the Tree-SID T-SID1 as Replication-SID. For SRv6, assume the Replication SID at node k, bound to an End.Replcate function, is 2001:db8:cccc:k:FA::/128.

A.1. P2MP Tree with non-adjacent Replication Segments

Assume PCE computes a P2MP tree with Root node R1, Intermediate and Leaf node R2, and Leaf nodes R6 and R7. The PCE instantiates the P2MP tree by stitching Replication segments at R1, R2, R6 and R7. Replication segment at R1 replicates to R2. Replication segment at R2 replicates to R6 and R7. Note nodes R3, R4 and R5 do not have any Replication segment state for the tree.

A.1.1. SR-MPLS

The Replication segment state at nodes R1, R2, R6 and R7 is shown below.

Replication segment at R1:

```
Replication segment <R1,T-ID,R1>:
 Replication SID: T-SID1
 Replication State:
   R2: <T-SID1->L12>
```

Replication to R2 steers packet directly to the node on interface L12.

Replication segment at R2:

```
Replication segment <R1,T-ID,R2>:
 Replication SID: T-SID1
 Replication State:
   R2: <Leaf>
   R6: <N-SID6, T-SID1>
   R7: <N-SID7, T-SID1>
```

R2 is a Bud-Node. It performs role of Leaf as well as a transit node replicating to R6 and R7. Replication to R6, using N-SID6, steers packet via IGP shortest path to that node. Replication to R7, using N-SID7, steers packet via IGP shortest path to R7 via either R5 or R4 based on ECMP hashing.

Replication segment at R6:

```
Replication segment <R1,T-ID,R6>:
 Replication SID: T-SID1
 Replication State:
   R6: <Leaf>
```

Replication segment at R7:

```
Replication segment <R1,T-ID,R7>:
 Replication SID: T-SID1
 Replication State:
   R7: <Leaf>
```

When a packet is steered into the SR P2MP Policy at R1:

- Since R1 is directly connected to R2, R1 performs PUSH operation with just <T-SID1> label for the replicated copy and sends it to R2 on interface L12.

- R2, as Leaf, performs NEXT operation, pops T-SID1 label and delivers the payload. For replication to R6, R2 performs a PUSH operation of N-SID6, to send <N-SID6,T-SID1> label stack to R3. R3 is the penultimate hop for N-SID6; it performs penultimate hop popping, which corresponds to the NEXT operation and the packet is then sent to R6 with <T-SID1> in the label stack. For replication to R7, R2 performs a PUSH operation of N-SID7, to send <N-SID7,T-SID1> label stack to R4, one of IGP ECMP nexthops towards R7. R4 is the penultimate hop for N-SID6; it performs penultimate hop popping, which corresponds to the NEXT operation and the packet is then sent to R7 with <T-SID1> in the label stack.

- R6, as Leaf, performs NEXT operation, pops T-SID1 label and delivers the payload.

- R7, as Leaf, performs NEXT operation, pops R-SID7 label and delivers the payload.

A.1.2. SRv6

For SRv6, the replicated packet from R2 to R7 has to traverse R4 using a SR-TE policy, Policy27. The policy has one SID in segment

list: End.X function with USD of R4 to R7 . The Replication segment state at nodes R1, R2, R6 and R7 is shown below.

```
Policy27: <2001:db8:cccc:4:C17::>
```

Replication segment at R1:

```
Replication segment <R1,T-ID,R1>:
 Replication SID: 2001:db8:cccc:1:FA::
 Replication State:
   R2: <2001:db8:cccc:2:FA::->L12>
```

Replication to R2 steers packet directly to the node on interface L12.

Replication segment at R2:

```
Replication segment <R1,T-ID,R2>:
 Replication SID: 2001:db8:cccc:2:FA::
 Replication State:
   R2: <Leaf>
   R6: <2001:db8:cccc:6:FA::>
   R7: <2001:db8:cccc:7:FA:: -> Policy27>
```

R2 is a Bud-Node. It performs role of Leaf as well as a transit node replicating to R6 and R7. Replication to R6, steers packet via IGP shortest path to that node. Replication to R7, via SR-TE policy, first encapsulates the packet using H.Encaps and then steers the outer packet to R4. End.X USD on R4 decapsulates outer header and sends the original inner packet to R7.

Replication segment at R6:

```
Replication segment <R1,T-ID,R6>:
 Replication SID: 2001:db8:cccc:6:FA::
 Replication State:
   R6: <Leaf>
```

Replication segment at R7:

```
Replication segment <R1,T-ID,R7>:
 Replication SID: 2001:db8:cccc:7:FA::
 Replication State:
   R7: <Leaf>
```

When a packet (A,B2) is steered into the SR P2MP Policy at R1 using H.Encaps.Replicate behavior:

*Since R1 is directly connected to R2, R1 sends replicated copy (2001:db8::1, 2001:db8:cccc:2:FA::) (A,B2) to R2 on interface L12.

*R2, as Leaf removes outer IPv6 header and delivers the payload. R2, as a bud node, also replicates the packet.

* -For replication to R6, R2 sends (2001:db8::1, 2001:db8:cccc:6:FA::) (A,B2) to R3. R3 forwards the packet using 2001:db8:cccc:6::/64 packet to R6.

-For replication to R7 using Policy27, R2 encapsulates and sends (2001:db8::2, 2001:db8:cccc:4:C17::) (2001:db8::1, 2001:db8:cccc:7:FA::) (A,B2) to R4. R4 performs End.X USD behavior, decapsulates outer IPv6 header and sends (2001:db8::1, 2001:db8:cccc:7:FA::) (A,B2) to R7.

*R6, as Leaf, removes outer IPv6 header and delivers the payload.

*R7, as Leaf, removes outer IPv6 header and delivers the payload.

A.2. P2MP Tree with adjacent Replication Segments

Assume PCE computes a P2MP tree with Root node R1, Intermediate and Leaf node R2, Intermediate nodes R3 and R5, and Leaf nodes R6 and R7. The PCE instantiates the P2MP tree by stitching Replication segments at R1, R2, R3, R5, R6 and R7. Replication segment at R1 replicates to R2. Replication segment at R2 replicates to R3 and R5. Replication segment at R3 replicates to R6. Replication segment at R5 replicates to R7. Note node R4 does not have any Replication segment state for the tree.

A.2.1. SR-MPLS

The Replication segment state at nodes R1, R2, R3, R5, R6 and R7 is shown below.

Replication segment at R1:

```
Replication segment <R1,T-ID,R1>:
 Replication SID: T-SID1
 Replication State:
   R2: <T-SID1->L12>
```

Replication to R2 steers packet directly to the node on interface L12.

```
   Replication segment at R2:

Replication segment <R1,T-ID,R2>:
 Replication SID: T-SID1
 Replication State:
   R2: <Leaf>
   R3: <T-SID1->L23>
   R5: <T-SID1->L25>

   R2 is a Bud-Node. It performs role of Leaf as well as a transit node
   replicating to R3 and R5. Replication to R3, steers packet directly
   to the node on L23. Replication to R5, steers packet directly to the
   node on L25.

   Replication segment at R3:

Replication segment <R1,T-ID,R3>:
 Replication SID: T-SID1
 Replication State:
   R6: <T-SID1->L36>

   Replication to R6, steers packet directly to the node on L36.

   Replication segment at R5:

Replication segment <R1,T-ID,R5>:
 Replication SID: T-SID1
 Replication State:
   R7: <T-SID1->L57>

   Replication to R7, steers packet directly to the node on L57.

   Replication segment at R6:

Replication segment <R1,T-ID,R6>:
 Replication SID: T-SID1
 Replication State:
   R6: <Leaf>

   Replication segment at R7:

Replication segment <R1,T-ID,R7>:
 Replication SID: T-SID1
 Replication State:
   R7: <Leaf>
```

When a packet is steered into the SR P2MP Policy at R1:

*Since R1 is directly connected to R2, R1 performs PUSH operation with just <T-SID1> label for the replicated copy and sends it to R2 on interface L12.

*R2, as Leaf, performs NEXT operation, pops T-SID1 label and delivers the payload. It also performs PUSH operation on T-SID1 for replication to R3 and R5. For replication to R6, R2 sends <T-SID1> label stack to R3 on interface L23. For replication to R5, R2 sends <T-SID1> label stack to R5 on interface L25.

*R3 performs NEXT operation on T-SID1 and performs a PUSH operation for replication to R6 and sends <T-SID1> label stack to R6 on interface L36.

*R5 performs NEXT operation on T-SID1 and performs a PUSH operation for replication to R7 and sends <T-SID1> label stack to R7 on interface L57.

*R6, as Leaf, performs NEXT operation, pops T-SID1 label and delivers the payload.

*R7, as Leaf, performs NEXT operation, pops R-SID7 label and delivers the payload.

A.2.2. SRv6

The Replication segment state at nodes R1, R2, R3, R5, R6 and R7 is shown below.

Replication segment at R1:

```
Replication segment <R1,T-ID,R1>:
 Replication SID: 2001:db8:cccc:1:FA::
 Replication State:
   R2: <2001:db8:cccc:2:FA::->L12>
```

Replication to R2 steers packet directly to the node on interface L12.

Replication segment at R2:

```
Replication segment <R1,T-ID,R2>:
 Replication SID: 2001:db8:cccc:2:FA::
 Replication State:
   R2: <Leaf>
   R3: <2001:db8:cccc:3:FA::->L23>
   R5: <2001:db8:cccc:5:FA::->L25>
```

R2 is a Bud-Node. It performs role of Leaf as well as a transit node replicating to R3 and R5. Replication to R3, steers packet directly to the node on L23. Replication to R5, steers packet directly to the node on L25.

Replication segment at R3:

```
Replication segment <R1,T-ID,R3>:
 Replication SID: 2001:db8:cccc:3:FA::
 Replication State:
   R6: <2001:db8:cccc:6:FA::->L36>
```

Replication to R6, steers packet directly to the node on L36.

Replication segment at R5:

```
Replication segment <R1,T-ID,R5>:
 Replication SID: 2001:db8:cccc:5:FA::
 Replication State:
   R7: <2001:db8:cccc:7:FA::->L57>
```

Replication to R7, steers packet directly to the node on L57.

Replication segment at R6:

```
Replication segment <R1,T-ID,R6>:
 Replication SID: 2001:db8:cccc:6:FA::
 Replication State:
   R6: <Leaf>
```

Replication segment at R7:

```
Replication segment <R1,T-ID,R7>:
 Replication SID: 2001:db8:cccc:7:FA::
 Replication State:
   R7: <Leaf>
```

When a packet (A,B2) is steered into the SR P2MP Policy at R1 using H.Encaps.Replicate behavior:

*Since R1 is directly connected to R2, R1 sends replicated copy (2001:db8::1, 2001:db8:cccc:2:FA::) (A,B2) to R2 on interface L12.

*R2, as Leaf, removes outer IPv6 header and delivers the payload. R2, as a bud node, also replicates the packet. For replication to R3, R2 sends (2001:db8::1, 2001:db8:cccc:3:FA::) (A,B2) to R3 on interface L23. For replication to R5, R2 sends (2001:db8::1, 2001:db8:cccc:5:FA::) (A,B2) to R5 on interface L25.

*R3 replicates and sends (2001:db8::1, 2001:db8:cccc:6:FA::)
(A,B2) to R6 on interface L36.

*R5 replicates and sends (2001:db8::1, 2001:db8:cccc:7:FA::)
(A,B2) to R7 on interface L57.

*R6, as Leaf, removes outer IPv6 header and delivers the payload.

*R7, as Leaf, removes outer IPv6 header and delivers the payload.

Authors' Addresses

Daniel Voyer (editor)
Bell Canada
Montreal
Canada

Email: daniel.voyer@bell.ca

Clarence Filsfils
Cisco Systems, Inc.
Brussels
Belgium

Email: cfilsfil@cisco.com

Rishabh Parekh
Cisco Systems, Inc.
San Jose,
United States of America

Email: riparekh@cisco.com

Hooman Bidgoli
Nokia
Ottawa
Canada

Email: hooman.bidgoli@nokia.com

Zhaohui Zhang
Juniper Networks

Email: zzhang@juniper.net