Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 February 2026

R. Parekh, Ed.
Arrcus
D. Voyer, Ed.
C. Filsfils
Cisco Systems, Inc.
H. Bidgoli
Nokia
Z. Zhang
Juniper Networks

4 August 2025

Segment Routing Point-to-Multipoint Policy
draft-ietf-pim-sr-p2mp-policy-15

Abstract

Point-to-Multipoint (P2MP) policy enables creation of P2MP trees for efficient multi-point packet delivery in a Segment Routing (SR) domain. A SR P2MP Policy consists of Candidate Paths (CP) which define the topology of P2MP tree instances in each Candidate Path. A P2MP tree instance is instantiated by a set of Replication segments.

This document specifies the architecture, signaling, and procedures for SR P2MP Policies within Segment Routing over MPLS (SR-MPLS) and Segment Routing over IPv6 (SRv6) dataplane. It defines the P2MP Policy construct, the roles of the root and leaf nodes, Candidate Paths and how P2MP trees using Replication Segments are instantiated and maintained. Additionally, it describes the required extensions for a controller to support P2MP path computation and provisioning.

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 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/.

Copyright Notice

Table of Contents

1. Introduction 3
1.1. Terminology 3
2. SR P2MP Policy 4
2.1. SR P2MP Policy Identification 4
2.2. Components of an SR P2MP Policy 4
2.3. Candidate Paths and P2MP Tree Instances 5
2.4. Steering traffic into a SR P2MP policy 5
3. P2MP Tree 6
3.1. Tree-SID and Replication segments 6
3.2. Shared Replication segments 6
3.3. Leaf Nodes in SR-MPLS 6
3.4. Packet forwarding in P2MP tree 7
4. Using a controller to build a P2MP Tree 7
4.1. SR P2MP Policy Provisioning on a controller 7
4.2. Controller Functions 8
4.3. P2MP Tree Compute 8
4.4. Instantiating P2MP tree on nodes 8
4.5. Protection 9
4.5.1. Local Protection 9
4.5.2. Path Protection 9
5. IANA Considerations 9
6. Security Considerations 9
7. Acknowledgements 10
8. Contributors 10
9. References 11

9.1. Normative References 11
9.2. Informative References 12
Appendix A. Illustration of SR P2MP Policy and P2MP Tree 12
A.1. P2MP Tree with non-adjacent Replication Segments 14
A.1.1. SR-MPLS 14
A.1.2. SRv6 16
A.2. P2MP Tree with adjacent Replication Segments 17
A.2.1. SR-MPLS 17
A.2.2. SRv6 19
Authors' Addresses 21

1. Introduction

A Multi-point service delivery can be realized with P2MP trees in a Segment Routing domain. A P2MP tree spans from a Root node to a set of Leaf nodes via intermediate Replication nodes. It consists of a Replication segment [RFC9524] at the root node, stitched to one or more Replication segments at Leaf nodes and intermediate Replication nodes. A Bud node [RFC9524] is a node that is both a Replication node and a Leaf node. Any mention of "Leaf node(s)" in this document should be considered as referring to "Leaf or Bud node(s)".

A Segment Routing P2MP Policy defines the Root and Leaf nodes of a P2MP tree. It has one or more Candidate Paths (CP) with optional constraints and/or optimization objectives.

A controller computes P2MP tree instances, from the Root to Leaf nodes, of a Candidate Path using the constraints and objectives specified in the Candidate Path. Once computed, the controller instantiates a P2MP tree instance in the SR domain by signaling Replication segments to the Root, Replication and Leaf nodes. A Path Computation Element (PCE) [RFC4655] is one example of such a controller.

The Replication segments of a P2MP tree can be instantiated for both SR-MPLS and SRv6 dataplanes, enabling efficient packet replication within an SR domain.

1.1. Terminology

This section defines terms used frequently in this document. Refer to Terminology section of [RFC9524] for definition of Replication segment and other terms associated with it.

SR P2MP Policy: A SR P2MP Policy is a mechanism to construct a P2MP tree in a SR domain by specifying a Root and Leaf nodes.

Candidate Path: A Candidate Path of SR P2MP Policy defines topological or resource constraints and optimization objectives that are used to construct a P2MP Tree instances.

P2MP Tree Instance: A P2MP tree instance in a SR domain is constructed by stitching Replication segments between Root and Leaf nodes of a SR P2MP Policy. A P2MP tree belongs to a Candidate Path and its topology is determined by constraints and optimization objective of the Candidate Path. This document uses terms P2MP tree instance and P2MP tree interchangeably.

2. SR P2MP Policy

An SR P2MP Policy is used to instantiate P2MP trees between a Root and Leaf nodes in an SR domain. It is similar to SR Policy [RFC9256]. Like SR Policy, SR P2MP Policy has one or more Candidate Paths and uses same criteria to select the Active Candidate Path. However, SR P2MP Policies and SR Policies are independent and distinct entities in a SR domain.

2.1. SR P2MP Policy Identification

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

* Root: The IP address of the Root node of P2MP trees instantiated by the SR P2MP Policy. This is equivalent to the Headend of SR Policy identifier tuple.

* Tree-ID: A 32-bit unsigned integer that uniquely identifies the P2MP Policy in the context of the Root node. This is equivalent to the Color of SR Policy identifier tuple.

Note, SR P2MP Policy identification tuple does not have a member equivalent to Endpoint of SR Policy identifier tuple since SR P2MP Policies result in P2MP trees to a varying set of endpoints (Leaf nodes).

2.2. Components of an SR P2MP Policy

An SR P2MP Policy consists of the following elements:

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

* Candidate Paths: A set of possible paths that define constraints and optimization objectives of P2MP trees of P2MP Policy.

A SR P2MP Policy is provisioned on a controller. The controller computes the P2MP tree instances of Candidate Paths of the policy and instantiates the necessary Replication segments at the Root, Replication and Leaf nodes of the trees. The Root and Tree-ID of the SR P2MP Policy are mapped to Replication-ID element of the Replication segment identifier, as specified[RFC9524].

2.3. Candidate Paths and P2MP Tree Instances

An SR P2MP Policy has one or more CPs. Identification of a CP in context of the P2MP Policy is as specified in [RFC9256]. A CP may include topological and/or resource constraints and optimization objectives which influence the computation of P2MP tree. The Root node selects the active Candidate Path based on the tie breaking rules defined in[RFC9256].

A Candidate Path has zero or more P2MP tree instances. A P2MP tree instance is identified by an Instance-ID. This is an unsigned 16-bit number which is unique in context of the SR P2MP Policy of the Candidate Path.

The Replication segments used to instantiate a P2MP tree instance are identified by the tuple: <Root, Tree-ID, Instance-ID, Node-ID>, where Root, Tree-ID of SR P2MP Policy and Instance-ID of the instance map to Replication-ID of Replication segment and Node-ID is as defined in [RFC9524].

The controller designates an active instance of a CP at the Root node of SR P2MP Policy by signalling this state through the protocol used to instantiate the Replication segment of the instance.

2.4. Steering traffic into a SR P2MP policy

The Tree-SID, as described in Section 3, serves as the forwarding plane identifier of a P2MP tree instance. It is instantiated in the forwarding plane at the Root node, intermediate Replication nodes, and Leaf nodes of the P2MP tree instance. The Tree-SID of the active instance of the active Candidate Path SHOULD be used as the Binding SID of the SR P2MP Policy.

The Root node can steer an incoming packet into a SR P2MP Policy in one of following methods:

* Local Policy-Based Routing: The Root node selects the active P2MP tree instance of the active Candidate Path of the SR P2MP Policy based on local policy. The procedures to map an incoming packet to a SR P2MP Policy are out of scope of this document.

* Tree-SID Based Routing: The Binding SID (Tree-SID) in the incoming packet is used to map the packet to the appropriate P2MP tree instance.

3. P2MP Tree

A P2MP tree within an SR domain establishes a forwarding structure that connects a Root node to a set of Leaf nodes via a series of intermediate Replication nodes. The tree consists of:

* A Replication segment at the Root node.

* Zero or more Replication Segments at intermediate Replication nodes.

* Replication Segments at the Leaf nodes.

3.1. Tree-SID and Replication segments

The Replication SID associated with the Replication segment at the Root node is referred to as the Tree-SID. The Tree-SID SHOULD also serve as the Replication-SID for the Replication segments at intermediate Replication nodes and Leaf nodes. However, the Replication Segments at intermediate Replication nodes and Leaf nodes MAY use Replication-SIDs that differ from the Tree-SID.

3.2. Shared Replication segments

A Replication segment MAY be shared across different P2MP tree instances, for example, in scenarios involving protection mechanisms. A shared Replication Segment SHOULD be identified using a Root-ID set to zero (0.0.0.0 for IPv4 and :: for IPv6) along with a Replication-ID that is unique within the context of the node where the Replication segment is instantiated.

However, a shared Replication segment MUST NOT be associated with an SR P2MP tree.

3.3. Leaf Nodes in SR-MPLS

For multi-point services, the transport identifier which is the Tree-SID or Replication SID at a Leaf node is also associated with the service context because it is not always feasible to separate the transport and service context with efficient replication in core since a) multi-point services may have differing sets of end-points, and b) downstream allocation of service context cannot be encoded in packets replicated in the core.

However, for SR-MPLS deployments, if it is known a priori that multi-point services mapped to a P2MP tree can be uniquely identified within the SR domain, a controller MAY opt not to instantiate Replication Segments at Leaf nodes. In such cases, Replication Nodes upstream of the Leaf nodes effectively implement Penultimate-Hop Popping behavior by removing the Tree-SID from the packet before forwarding it. A multi-point service context allocated from the Domain-wide Common Block (DCB), as specified in [RFC9573], is an example of a globally unique context that facilitates this optimization.

3.4. Packet forwarding in P2MP tree

When a packet is steered into a P2MP tree instance, the Replication segment at the Root node performs packet replication and forwards copies to downstream nodes.

* Each replicated packet carries the Replication-SID of the Replication segment at the downstream node.

* A downstream node can be either:

  - A Leaf node, in which case the replication process terminates.

  - An intermediate Replication node, which further replicates the packet through its associated Replication segments until it reaches all Leaf nodes.

4. Using a controller to build a P2MP Tree

A controller is provisioned with SR P2MP Policy and it’s Candidate Paths to compute and instantiate P2MP trees in SR domain. Once computed, the controller instantiates the Replication segments that compose the P2MP in the SR domain nodes using signalling protocols such as PCEP, BGP, NetConf etc. The procedures for provisioning a controller and the instantiation Replication segments in SR domain are outside the scope of this document.

4.1. SR P2MP Policy Provisioning on a controller

An entity (an operator, a network node or a machine) provisions a SR P2MP Policy on a controller by specifying the addresses of the Root, set of Leaf nodes Candidate Paths. The procedures and mechanisms for provisioning a controller are outside the scope of this document.

Candidate Path constraints MAY include link color affinity, bandwidth, disjointness (link, node, SRLG), delay bound, link loss, flexible algorithm etc., and optimization objectives based on IGP or TE metric or link latency. Other constraints and optimization objectives MAY be used for P2MP tree computation.

The Tree SID of a P2MP Tree instance of a Candidate Path of a SR P2MP Policy can be either dynamically assigned by the controller or statically assigned by entity provisioning the SR P2MP Policy.

4.2. Controller Functions

A controller performs the following functions in general:

* Topology Discovery: A controller discovers network topology across Interior Gateway Protocol (IGP) areas, levels or Autonomous Systems (ASs).

* Capability Exchange: A controller discovers a node’s capability to participate in SR P2MP tree as well as advertise it’s capability to compute P2MP trees.

4.3. P2MP Tree Compute

A controller computes one or more P2MP tree instances for CPs of a SR P2MP Policy. A CP may not have any P2MP tree instance if a controller cannot compute a P2MP tree instance for it..

A controller MUST compute a P2MP tree such that there are no loops in the tree at steady state as required by [RFC9524]).

A controller SHOULD modify a P2MP tree of a Candidate Path on detecting a change in the network topology or in case a better path can be found based on the new network state. In this case, the controller MAY create a new instance of a P2MP tree and remove the old instance of the tree from the network in order to minimize traffic loss.

4.4. Instantiating P2MP tree on nodes

Once a controller computes a P2MP tree instance for a CP of a SR P2MP Policy, it needs to instantiate the tree on the relevant network nodes via Replication segments. The controller can use various mechanisms to program the Replication segments as described below. Some examples of these mechanisms are PCEP, BGP and NetConf. The mechanism and procedures to signal Replication segments are outside the scope of this document.

4.5. Protection

4.5.1. Local Protection

A network link, node or path on the instance of a P2MP tree can be protected using SR policies computed by a controller. The backup SR policies are programmed in forwarding plane in order to minimize traffic loss when the protected link/node fails.

It is also possible to use node local Loop-Free Alternate protection and Micro-Loop Prevention mechanisms to protect link/nodes of P2MP tree.

4.5.2. Path Protection

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

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

This document describes how a P2MP tree can be created in an SR domain by stitching Replication Segments together. Some security considerations for Replication Segments outlined in [RFC9524] are also applicable to this document. Following is a brief reminder of the same.

An SR domain needs protection from outside attackers as described in [RFC8754].

Failure to protect the SR MPLS domain by correctly provisioning MPLS support per interface permits attackers from outside the domain to send packets to receivers of the Multi-point services that use the SR P2MP trees provisioned within the domain.

Failure to protect the SRv6 domain with inbound Infrastructure Access Control Lists (IACLs) on external interfaces, combined with failure to implement BCP 38 [RFC2827] or apply IACLs on nodes provisioning SIDs, permits attackers from outside the SR domain to send packets to the receivers of Multi-point services that use the SR P2MP trees provisioned within the domain.

Incorrect provisioning of Replication segments by a controller that computes SR P2MP tree instance can result in a chain of Replication segments forming a loop. In this case, replicated packets can create a storm till MPLS TTL (for SR-MPLS) or IPv6 Hop Limit (for SRv6) decrements to zero.

The control plane protocols (like PCEP, BGP, etc.) used to instantiate Replication segments of SR P2MP tree instance can leverage their own security mechanisms such as encryption, authentication filtering etc.

For SRv6, [RFC9524] describes an exception for Parameter Problem Message, code 2 ICMPv6 Error messages. If an attacker is able to inject a packet into Multi-point service with source address of a node and with an extension header using unknown option type marked as mandatory, then a large number of ICMPv6 Parameter Problem messages can cause a denial-of-service attack on the source node.

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

Kamran Raza Cisco Systems, Inc. Canada

Email:skraza@cisco.com

Anuj Budhiraja Cisco Systems, Inc. US

Email:abudhira@cisco.com

Mankamana Mishra Cisco Systems, Inc. US

Email:mankamis@cisco.com

9. References

9.1. Normative References

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

[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>.

[RFC9256] Filsfils, C., Talaulikar, K., Ed., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", RFC 9256, DOI 10.17487/RFC9256, July 2022, <https://www.rfc-editor.org/info/rfc9256>.

[RFC9524] Voyer, D., Ed., Filsfils, C., Parekh, R., Bidgoli, H., and Z. Zhang, "Segment Routing Replication for Multipoint Service Delivery", RFC 9524, DOI 10.17487/RFC9524, February 2024, <https://www.rfc-editor.org/info/rfc9524>.

9.2. Informative References

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

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <https://www.rfc-editor.org/info/rfc2827>.

[RFC4655] Farrel, A., Vasseur, J.-P., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <https://www.rfc-editor.org/info/rfc4655>.

[RFC8754] Filsfils, C., Ed., Dukes, D., Ed., Previdi, S., Leddy, J., Matsushima, S., and D. Voyer, "IPv6 Segment Routing Header (SRH)", RFC 8754, DOI 10.17487/RFC8754, March 2020, <https://www.rfc-editor.org/info/rfc8754>.

[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>.

[RFC9573] Zhang, Z., Rosen, E., Lin, W., Li, Z., and IJ. Wijnands, "MVPN/EVPN Tunnel Aggregation with Common Labels", RFC 9573, DOI 10.17487/RFC9573, May 2024, <https://www.rfc-editor.org/info/rfc9573>.

Appendix A. Illustration of SR P2MP Policy and P2MP Tree

Consider the following topology:

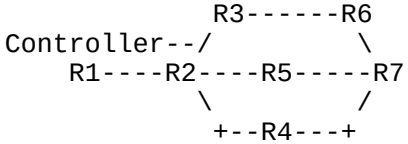

Figure 1: Figure 1

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. This document re-uses SID allocation scheme, reproduced below, from Illustrations in 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 a controller is provisioned with 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 controller is responsible for computing a P2MP tree instance of the Candidate Path. In this example, we assume one active instance of P2MP tree with Instance-ID I-ID1. Assume the controller instantiates P2MP trees by signalling Replication segments i.e. Replication-ID of these Replication segments is <Root, Tree-ID, Instance-ID>.. 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.Replicate function, is 2001:db8:cccc:k:FA::/128.

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

Assume the controller computes a P2MP tree instance with Root node R1, Intermediate and Leaf node R2, and Leaf nodes R6 and R7. The controller instantiates the instance 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,I-ID1,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,I-ID1,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,I-ID1,R6>:
 Replication-SID: T-SID1
 Replication State:
   R6: <Leaf>
```

Replication segment at R7:

```
Replication segment <R1,T-ID,I-ID1,R7>:
 Replication-SID: T-SID1
 Replication State:
   R7: <Leaf>
```

When a packet is steered into the active instance Candidate Path 1 of 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-SID7; 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 T-SID1 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,I-ID1,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,I-ID1,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,I-ID1,R6>:
 Replication-SID: 2001:db8:cccc:6:FA::
 Replication State:
   R6: <Leaf>

Replication segment at R7:

Replication segment <R1,T-ID,I-ID1,R7>:
 Replication-SID: 2001:db8:cccc:7:FA::
 Replication State:
   R7: <Leaf>
```

When a packet (A,B2) is steered into the active instance of Candidate Path 1 of 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 the controller 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 controller instantiates the P2MP tree instance 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,I-ID1,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,I-ID1,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,I-ID1,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,I-ID1,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,I-ID1,R6>:
 Replication-SID: T-SID1
 Replication State:
   R6: <Leaf>

Replication segment at R7:

Replication segment <R1,T-ID,I-ID1,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 T-SID1 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,I-ID1,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,I-ID1,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,I-ID1,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,I-ID1,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,I-ID1,R6>:
 Replication-SID: 2001:db8:cccc:6:FA::
 Replication State:
   R6: <Leaf>

Replication segment at R7:

Replication segment <R1,T-ID,I-ID1,R7>:
 Replication-SID: 2001:db8:cccc:7:FA::
 Replication State:
   R7: <Leaf>

When a packet (A,B2) is steered into the active instance of Candidate Path 1 of 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

Rishabh Parekh (editor)
Arrcus
San Jose,
United States of America
Email: rishabh@arrcus.com

Daniel Voyer (editor)
Cisco Systems, Inc.
Montreal
Canada
Email: davoyer@cisco.com

Clarence Filsfils
Cisco Systems, Inc.
Brussels
Belgium
Email: cfilsfil@cisco.com

Hooman Bidgoli
Nokia
Ottawa
Canada
Email: hooman.bidgoli@nokia.com

Zhaohui Zhang
Juniper Networks
Email: zzhang@juniper.net