Stream: Internet Engineering Task Force (IETF)
RFC: 9960
Updates: 9524
Category: Standards Track
Published: April 2026
ISSN: 2070-1721
Authors:
R. Parekh, Ed. *Arrcus*
D. Voyer, Ed. *Cisco Systems, Inc.*
C. Filsfils *Cisco Systems, Inc.*
H. Bidgoli *Nokia*
Z. Zhang *Juniper Networks*

RFC 9960
Segment Routing Point-to-Multipoint Policy

Abstract

The Point-to-Multipoint (P2MP) Policy enables creation of P2MP trees for efficient multipoint packet delivery in a Segment Routing (SR) domain. This document specifies the architecture, signaling, and procedures for SR P2MP Policies with Segment Routing over MPLS (SR-MPLS) and Segment Routing over IPv6 (SRv6). It defines the SR P2MP Policy construct, candidate paths (CPs) of an SR P2MP Policy, and the instantiation of the P2MP tree instances (PTIs) of a CP using Replication segments. Additionally, it describes the required extensions for a controller to support P2MP path computation and provisioning. This document updates RFC 9524.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

[RFC9524] defines a Replication segment that enables a SR node to replicate traffic to multiple downstream nodes in an SR domain [RFC8402]. A Point-to-Multipoint (P2MP) service can be realized by a single Replication segment spanning from the ingress node to the egress nodes of the service. This effectively achieves ingress replication, which is inefficient since the traffic of the P2MP service may traverse the same set of nodes and links in the SR domain on its path from the ingress node to the egress nodes.

A multipoint service delivery can be efficiently realized with a P2MP tree in a SR 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 at the Root node, and that Replication segment is stitched to one or more Replication segments between the 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)".

An SR P2MP Policy defines the Root and Leaf nodes of a P2MP tree. It has one or more CPs provisioned with optional constraints and/or optimization objectives.

A controller computes PTIs of the CPs using the constraints and objectives specified in the CP. The controller then instantiates a PTI 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. In other cases, a PTI can be installed using the Network Configuration Protocol (NETCONF) / YANG or the Command Line Interface (CLI) on the Root, Replication, and Leaf nodes.

The Replication segments of a PTI can be instantiated for SR-MPLS [RFC8660] and SRv6 [RFC8986] data planes, enabling efficient packet replication within an SR domain.

This document updates the Replication-ID portion of the Replication segment identifier (Replication-SID) specified in Section 2 of [RFC9524].

1.1. Terminology

This section defines terms used frequently in this document. Refer to the Terminology section of [RFC9524] for the definitions of Replication segment and other terms associated with it and the definitions of Root, Leaf, and Bud nodes.

SR P2MP Policy: An SR P2MP Policy is a framework to construct P2MP trees in an SR domain by specifying Root and Leaf nodes.

Tree-ID: An identifier of an SR P2MP Policy in context of the Root node.

Candidate path (CP): A CP of the SR P2MP Policy defines topological or resource constraints and optimization objectives that are used to compute and construct PTIs.

P2MP tree instance (PTI): A PTI of a CP is constructed by stitching Replication segments between the Root and Leaf nodes of an SR P2MP Policy. Its topology is determined by the constraints and optimization objective of the CP.

Instance-ID: An identifier of a PTI in context of the SR P2MP Policy.

Tree-SID: The Replication-SID of the Replication segment at the Root node of a PTI.

1.2. 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.

2. SR P2MP Policy

An SR P2MP Policy is used to instantiate P2MP trees between Root and Leaf nodes in an SR domain. Note that multiple SR P2MP Policies can have identical Root nodes and identical sets of Leaf nodes. An SR P2MP Policy has one or more CPs [RFC9256].

2.1. SR P2MP Policy Identification

An 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.
- Tree-ID: A 32-bit unsigned integer that uniquely identifies the SR P2MP Policy in the context of the Root node.

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 SR P2MP Policy.
- Candidate paths: A set of possible paths that define constraints and optimization objectives for PTIs of the SR P2MP Policy.

An SR P2MP Policy and its CPs are provisioned on a controller (see Section 5) or the Root node or both, depending upon the provisioning model. After provisioning, the Policy and its CPs are instantiated on the Root node or the controller by using a signaling protocol.

2.3. Candidate Paths and P2MP Tree Instances

An SR P2MP Policy has one or more CPs. The tuple <Protocol-Origin, Originator, Discriminator>, as specified in Section 2.6 of [RFC9256], uniquely identifies a CP in the context of an SR P2MP Policy. The semantics of Protocol-Origin, Originator, and Discriminator fields of the identifier are the same as in Sections 2.3, 2.4, and 2.5 of [RFC9256], respectively.

The Root node of the SR P2MP Policy selects the active CP based on the tiebreaking rules defined in Section 2.9 of [RFC9256].

A CP may include topological and/or resource constraints and optimization objectives that influence the computation of the PTIs of the CP.

A CP has zero or more PTIs. A CP does not have a PTI when the controller cannot compute a P2MP tree from the network topology based on the constraints and/or optimization objectives of the CP. A CP can have more than one PTI, e.g., during the Make-Before-Break (see Section 5.3) procedure to handle a network state change. However, one and only one PTI **MUST** be the active instance of the CP. If more than one PTI of a CP is active at same time, and that CP is the active CP of the SR P2MP Policy, then duplicate traffic may be delivered to the Leaf nodes.

A PTI is identified by an Instance-ID. This is an unsigned 16-bit number that is unique in context of the SR P2MP Policy of the CP.

PTIs are instantiated using Replication segments. Section 2 of [RFC9524] specifies the Replication-ID of the Replication segment control plane identifier tuple as a variable length field that can be modified as required based on the use of a Replication segment. However, length is an imprecise indicator of the actual structure of the Replication-ID. This document updates the Replication-ID of the control plane identifier of a Replication segment [RFC9524] to be the tuple: <Root, Tree-ID, Instance-ID>, where <Root, Tree-ID> identifies the SR P2MP Policy and Instance-ID identifies the PTI within that SR P2MP Policy. The Replication segments used to instantiate a PTI are thus identified in the control plane by the tuple: <Root, Tree-ID, Instance-ID, Node-ID>. As per Section 2 of [RFC9524], for a simple use case, the Replication-ID is a 32-bit number. To map this use case to the tuple for the control plane identifier of a Replication segment as defined in this document, the Root **MUST** be zero (0.0.0.0 for IPv4 and :: for IPv6), the Instance-ID **MUST** be zero, and the 32-bit Tree-ID to effectively make the tuple <[0.0.0.0 or ::], Tree-ID, 0, Node-ID>.

PTIs may have different tree topologies due to possibly differing constraints and optimization objectives of the CPs in an SR P2MP Policy and across different policies. Even within a given CP, two PTIs of that CP, say during the Make-Before-Break procedure, are likely to have different tree topologies due to a change in the network state. Since the PTIs may have different tree topologies, their replication states also differ at various nodes in the SR domain. Therefore, each PTI has its own Replication segment and a unique Replication-SID in the data plane at a given node in the SR domain.

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

This document focuses on the use of a controller to compute and instantiate PTIs of SR P2MP Policy CPs. It is also feasible to provision an explicit CP in an SR P2MP Policy with a static tree topology using NETCONF/YANG or CLI. Note that a static tree topology will not adapt to any changes in the network state of an SR domain. The explicit CPs may be provisioned on the controller or the Root node. When an explicit CP is provisioned on the controller, the controller bypasses the compute stage and directly instantiates the PTIs in the SR domain. When an explicit CP is provisioned on the Root node, the Root node instantiates the PTIs in the SR domain. The exact procedures for provisioning an explicit CP and the signaling from the Root node to instantiate the PTIs are outside the scope of this document.

3. Steering Traffic into an SR P2MP Policy

The Replication-SID of the Replication segment at the Root node is referred to as the Tree-SID of a PTI. It is **RECOMMENDED** that the Tree-SID is also used as the Replication-SID for the Replication segments at the intermediate Replication nodes and the Leaf nodes of the PTI as it simplifies operations and troubleshooting. However, the Replication-SIDs of the Replication segments at the intermediate Replication nodes and the Leaf nodes **MAY** differ from the Tree-SID. For SRv6, Replication-SID is the FUNCT portion of the SRv6 segment ID (SID) [RFC8986] [RFC9524]. Note that even if the Tree-SID is the Replication-SID of all the Replication segments of a PTI, the locator (LOC) portion of the SRv6 SID [RFC8986] differs for the Root node, the intermediate Replication nodes, and the Leaf nodes of the PTI.

An SR P2MP Policy has a Binding SID (BSID). The BSID is used to steer traffic into an SR P2MP Policy, as described below, when the Root node is not the ingress node of the SR domain where the traffic arrives. The packets are steered from the ingress node to the Root node using a segment list with the BSID as the last segment in the list. In this case, it is **RECOMMENDED** that the BSID of an SR P2MP Policy **SHOULD** be constant throughout the lifetime of the policy so the steering of traffic to the Root node remains unchanged. The BSID of an SR P2MP Policy **MAY** be the Tree-SID of the active P2MP instance of the active CP of the policy. In this case, the BSID of an SR P2MP Policy changes when the active CP or the active PTI of the SR P2MP Policy changes. Note that the BSID is not required to steer traffic into an SR P2MP Policy when the Root node of an SR P2MP Policy is also the ingress node of the SR domain where the traffic arrives.

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

- Local-policy-based forwarding: The Root node maps the incoming packet to the active PTI of the active CP of an SR P2MP Policy based on local forwarding policy, and it is replicated with the encapsulated Replication-SIDs of the downstream nodes. The procedures to map an incoming packet to an SR P2MP Policy are out of scope of this document. It is **RECOMMENDED** that an implementation provide a mechanism to examine the result of application of the local forwarding policy, i.e., provide information about the traffic mapped to an SR P2MP Policy and the active CP and active PTI of the policy.
- Tree-SID-based forwarding: The BSID, which may be the Tree-SID of the active PTI, in an incoming packet is used to map the packet to the active PTI. The BSID in the incoming packet is replaced with the Tree-SID of the active PTI of the active CP, and the packet is replicated with the Replication-SIDs of the downstream nodes.

For local-policy-based forwarding with SR-MPLS, the TTL for the Root node **SHOULD** set the TTL in the encapsulating MPLS header so that the replicated packet can reach the furthest Leaf node. The Root **MAY** set the TTL in the encapsulating MPLS header from the payload. In this case, the TTL may not be sufficient for the replicated packet to reach the furthest node. For SRv6, Section 2.2 of [RFC9524] provides guidance to set the IPv6 Hop Limit of the encapsulating IPv6 header.

4. P2MP Tree Instance

A PTI 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.

4.1. Replication Segments at Leaf Nodes

A specific service is identified by a service context in a packet. A PTI is usually associated with one and only one multipoint service. On a Leaf node of such a multipoint service, the transport identifier, which is the Tree-SID or Replication-SID of the Replication segment 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) multipoint services may have differing sets of endpoints and b) downstream allocation of a service context cannot be encoded in packets replicated in the core.

A PTI can be associated with one or more multipoint services on the Root and Leaf nodes. In SR-MPLS deployments, if it is known a priori that multipoint services mapped to an SR-MPLS PTI can be uniquely identified with their service label, a controller **MAY** opt to not instantiate Replication segments at Leaf nodes. In such cases, Replication nodes upstream of the Leaf nodes can remove the Tree-SID from the packet before forwarding it. A multipoint service context allocated from an upstream assigned label or Domain-wide Common Block (DCB), as specified in [RFC9573], is an example of a globally unique context that facilitates this optimization.

In SRv6 deployments, Replication segments of a PTI **MUST** be instantiated on Leaf nodes of the tree since behavior like Penultimate Hop Popping (PHP) is not feasible because the Tree-SID is carried in the IPv6 Destination Address field of the outer IPv6 header. If two or more multipoint services are mapped to one SRv6 PTI, an SRV6 SID representing the service context is assigned by the Root node or assigned from the DCB. This SRv6 SID **MUST** be encoded as the last segment in the Segment List of the Segment Routing Header [RFC8754] by the Root node to derive the packet processing context (PPC) for the service, as described in Section 2.2 of [RFC9524], at a Leaf node.

4.2. Shared Replication Segments

A Replication segment **MAY** be shared across different PTIs. One simple use of a shared Replication segment is for local protection on a Replication node. Assume a Replication node, say node X, has multiple PTIs. Assume the Replications segments of these PTIs replicate to a downstream node, say node Y, amongst other downstream nodes. This node Y is a common downstream node of these Replication segments at node X. A Replication segment is established to protect the adjacency or path between node X and node Y; this Replication segment can be shared across all the Replication segments of the PTIs replicating from node X to node Y.

A shared Replication segment **MUST** be identified using a Root set to zero (0.0.0.0 for IPv4 and :: for IPv6), an Instance-ID set to zero, and a Tree-ID that is unique within the context of the node where the Replication segment is instantiated. The Root is zero because a shared Replication segment is not associated with a particular SR P2MP Policy or a PTI. Note that the shared Replication-SID conforms with the updated Replication-ID definition in Section 2.3.

It is possible for different PTIs to share a P2MP tree at a Replication node. This allows a common sub-tree to be shared across PTIs whose tree topologies are identical in some portion of an SR domain. The procedures to share a P2MP tree across PTIs are outside the scope of this document.

4.3. Packet Forwarding in a P2MP Tree Instance

When a packet is steered into a PTI, 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, or
  - An intermediate Replication node, which further replicates the packet through its associated Replication segments until it reaches all Leaf nodes.

A Replication node and a downstream node can be non-adjacent. In this case, the replicated packet has to traverse a path to reach the downstream node. For SR-MPLS, this is achieved by inserting one or more SIDs before the downstream Replication-SID. For SRv6, the LOC [RFC8986] of the downstream Replication-SID can guide the packet to the downstream node or an optional segment list may be used to steer the replicated packet on a specific path to the downstream node. For details of SRv6 replication to a non-adjacent downstream node and IPv6 Hop Limit considerations, refer to Section 2.2 of [RFC9524].

5. Using a Controller to Build a P2MP Tree

A controller is instantiated or provisioned with the SR P2MP Policy and its CPs to compute and instantiate PTIs in an SR domain. The procedures for provisioning or instantiation of these constructs on a controller are outside the scope of this document.

5.1. SR P2MP Policy on a Controller

An SR P2MP Policy is provisioned on a controller by an entity that can be an operator, a network node, or a machine by specifying the addresses of the Root, the set of Leaf nodes, and the CPs. In this case, the policy and its CPs are instantiated on the Root node using a signaling protocol. An SR P2MP Policy, its Leaf nodes, and the CPs may also be provisioned on the Root node and then instantiated on the controller using a signaling protocol. The procedures and mechanisms for provisioning and instantiation of an SR P2MP Policy and its CPS on a controller or a Root node are outside the scope of this document.

The possible set of constraints and optimization objective of a CP are described in Section 3 of [SR-POLICY]. Other constraints and optimization objectives **MAY** be used for P2MP tree computation.

5.2. 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 (ASes).
- Capability Exchange: A controller discovers a node's capability to participate in an SR P2MP Policy as well as advertise its capability to support the SR P2MP Policy.

5.3. P2MP Tree Compute

A controller computes one or more PTIs for CPs of an SR P2MP Policy. A CP may not have any PTIs if a controller cannot compute a P2MP tree 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 PTI of a CP on detecting a change in the network topology if the change affects the tree instance or when a better path can be found based on the new network state. Alternatively, the controller **MAY** decide to implement a Make-Before-Break approach to minimize traffic loss. The controller can do this by creating a new PTI, activating the new instance once it is instantiated in the network, and then removing the old PTI.

5.4. SID Management

The controller assigns the Replication-SIDs for the Replication segments of the PTI.

The Replication-SIDs of a PTI of a CP of an SR P2MP Policy can be either dynamically assigned by the controller or statically assigned by the entity provisioning the SR P2MP Policy.

For SR-MPLS, a Replication-SID may be assigned from the SR Local Block (SRLB) or the SR Global Block (SRGB) [RFC8402]. It is **RECOMMENDED** to assign a Replication-SID from the SRLB since Replication segments are local to each node of the PTI. It is **NOT RECOMMENDED** to allocate a Replication-SID from the SRGB since this block is globally significant in the SR domain any it may get depleted if a significant number of PTIs are instantiated in the SR domain.

Section 3 recommends that the Tree-SID be used as the Replication-SIDs for all the Replication segments of a PTI. It may be feasible to allocate the same Tree-SID value for all the Replication segments if the blocks used for allocation are not identical on all the nodes of the PTI or if the particular Tree-SID value in the block is assigned to some other SID on some node.

A BSID is also assigned for the SR P2MP Policy. The controller **MAY** decide to not assign a BSID and allow the Root node of the SR P2MP Policy to assign the BSID. It is **RECOMMENDED** to assign the BSID of an SR P2MP Policy from the SRLB for SR-MPLS.

The controller **MAY** be provisioned with a reserved block or multiple reserved blocks for assigning Replication-SIDs and/or the BSIDs for SR P2MP Policies. A single block maybe be reserved for the whole SR domain, or dedicated blocks can be reserved for each node or a group of nodes in the SR domain. These blocks **MAY** overlap with either the SRGB, the SRLB, or both. The procedures for provisioning these reserved blocks and procedures for deconflicting assignments from these reserved blocks with overlapping SRLB or SRGB blocks are outside the scope of this document.

A controller may not be aware of all the assignments of SIDs from the SRGB or the SRLB of the SR domain. If reserved blocks are not used, the assignment of Replication-SIDs or BSIDs of SR P2MP Policies from these blocks may conflict with other SIDs.

5.5. Instantiating P2MP Tree Instance on Nodes

After computing P2MP trees, the controller instantiates the Replication segments that compose the PTIs in the SR domain using signaling protocols such as the Path Computation Element Communication Protocol (PCEP) [SR-P2MP-PCEP], BGP [P2MP-BGP], or other mechanisms such as NETCONF/YANG [SR-P2MP-YANG], etc. The procedures for the instantiation of the Replication segments in an SR domain are outside the scope of this document.

A node **SHOULD** report a successful instantiation of a Replication segment. The exact procedure for reporting this is outside the scope of this document.

The instantiation of a Replication segment on a node may fail, e.g., when the Replication-SID conflicts with another SID on the node. The node **SHOULD** report this, preferably with a reason for the failure, using a signaling protocol. The exact procedure for reporting this failure is outside the scope of this document.

If the instantiation of a Replication segment on a node fails, the controller **SHOULD** attempt to re-instantiate the Replication segment. There **SHOULD** be an upper bound on the number of attempts. If the instantiation of a Replication segment ultimately fails after the allowed number of attempts, the controller **SHOULD** generate an alert via mechanisms like syslog. These alerts **SHOULD** be rate-limited to protect the logging facility in case Replication segment instantiation fails on multiple nodes. The controller **MAY** decide to tear down the PTI if the instantiations of some of the Replication segments of the instance fail. The controller is **RECOMMENDED** to tear down the PTI if the instantiation of the Replication segment on the Root node fails. The controller can employ different strategies to retry instantiating a PTI after a failure. These are out of scope of this document.

A PTI should be instantiated within a reasonable time, especially if it is the active PTI of an SR P2MP Policy. One approach is the controller instantiates the Replication segments in a batch. For example, the controller instantiates the Replication segments of the Leaf nodes and the intermediate Replication nodes first. If all of these Replication segments are successfully instantiated, the controller then proceeds to instantiate the Replication segment at the Root node. If the Replication segment instantiation at the Root node succeeds, the controller can immediately activate the instance if it needs to carry traffic of the SR P2MP Policy. A controller can adopt a similar approach when instantiating the new PTI for the Make-Before-Break procedure.

5.6. Protection

5.6.1. Local Protection

A network link, node, or replication branch on a PTI can be protected using SR Policies [RFC9256]. The backup SR Policies are associated with replication branches of a Replication segment and are programmed in the data plane in order to minimize traffic loss when the protected link/node fails. The segment list of the backup SR Policy is imposed on the downstream Replication-SID of a replication branch to steer the traffic on the backup path.

It is also possible to use a node local Loop-Free Alternate [RFC5286] or Topology Independent Loop-Free Alternate (TI-LFA) [RFC9855] protection and a Micro-Loop [RFC5715] or SR Micro-Loop [SR-LOOP] prevention mechanism to protect the links/nodes of a PTI.

5.6.2. Path Protection

A controller can create a disjoint backup tree instance for providing end-to-end tree protection if the topology permits. This can be achieved by having a backup CP with constraints and/or optimization objectives that ensure its PTIs are disjoint from the PTIs of the primary/active CP.

6. IANA Considerations

This document has no IANA actions.

7. Security Considerations

This document describes how a PTI 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 those security considerations.

An SR domain needs protection from outside attackers as described in [RFC8402], [RFC8754], and [RFC8986].

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 multipoint services that use the SR P2MP Policies 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 the method described in RFC 2827 [BCP38] or apply IACLs on nodes provisioning SIDs, permits attackers from outside the SR domain to send packets to the receivers of multipoint services that use the SR P2MP Policies provisioned within the domain.

Incorrect provisioning of Replication segments by a controller that computes SR PTI can result in a chain of Replication segments forming a loop. In this case, replicated packets can create a storm until 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 PTI can leverage their own security mechanisms such as encryption, authentication filtering, etc.

For SRv6, [RFC9524] describes an exception for the ICMPv6 Parameter Problem message with Code 2. If an attacker is able to inject a packet into a multipoint service with the source address of a node and with an extension header using an 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.

8. References

8.1. 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>.

[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>.

[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>.

8.2. Informative References

[BCP38] Best Current Practice 38, <https://www.rfc-editor.org/info/bcp38>. At the time of writing, this BCP comprises the following:

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>.

[P2MP-BGP] Bidgoli, H., Voyer, D., Stone, A., Parekh, R., Krier, S., and S. Agrewal, "Advertising p2mp policies in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-sr-p2mp-policy-01, 29 April 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-sr-p2mp-policy-01>.

[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>.

[RFC5286] Atlas, A., Ed. and A. Zinin, Ed., "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286, DOI 10.17487/RFC5286, September 2008, <https://www.rfc-editor.org/info/rfc5286>.

[RFC5715] Shand, M. and S. Bryant, "A Framework for Loop-Free Convergence", RFC 5715, DOI 10.17487/RFC5715, January 2010, <https://www.rfc-editor.org/info/rfc5715>.

[RFC8660] Bashandy, A., Ed., Filsfils, C., Ed., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing with the MPLS Data Plane", RFC 8660, DOI 10.17487/RFC8660, December 2019, <https://www.rfc-editor.org/info/rfc8660>.

[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>.

[RFC9855] Bashandy, A., Litkowski, S., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute Using Segment Routing", RFC 9855, DOI 10.17487/RFC9855, October 2025, <https://www.rfc-editor.org/info/rfc9855>.

[SR-LOOP] Bashandy, A., Filsfils, C., Litkowski, S., Decraene, B., Francois, P., and P. Psenak, "Loop avoidance using Segment Routing", Work in Progress, Internet-Draft, draft-bashandy-rtgwg-segment-routing-uloop-18, 19 April 2026, <https://datatracker.ietf.org/doc/html/draft-bashandy-rtgwg-segment-routing-uloop-18>.

[SR-P2MP-PCEP] Bidgoli, H., Voyer, D., Budhiraja, A., Parekh, R., and S. Sivabalan, "PCEP extensions for SR P2MP Policy", Work in Progress, Internet-Draft, draft-ietf-pce-sr-p2mp-policy-14, 23 February 2026, <https://datatracker.ietf.org/doc/html/draft-ietf-pce-sr-p2mp-policy-14>.

[SR-P2MP-YANG] Bidgoli, H., Voyer, D., Parekh, R., Saad, T., and T. Kundu, "YANG Data Model for p2mp sr policy", Work in Progress, Internet-Draft, draft-hb-spring-sr-p2mp-policy-yang-02, 30 October 2020, <https://datatracker.ietf.org/doc/html/draft-hb-spring-sr-p2mp-policy-yang-02>.

[SR-POLICY] Filsfils, C., Talaulikar, K., Król, P. G., Horneffer, M., and P. Mattes, "SR Policy Implementation and Deployment Considerations", Work in Progress, Internet-Draft, draft-filsfils-spring-sr-policy-considerations-09, 24 April 2022, <https://datatracker.ietf.org/doc/html/draft-filsfils-spring-sr-policy-considerations-09>.

Appendix A. Illustration of the SR P2MP Policy and P2MP Tree

Consider the following topology:

```
                 R3------R6
   Controller--/         \
           R1----R2----R5-----R7
                   \         /
                    +--R4---+
```

Figure 1: SR Topology

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

- 2001:db8::/32 is an IPv6 block allocated by a Regional Internet Registry (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 is expressed in 64 bits and a function is 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 Penultimate Segment Pop (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 Ultimate Segment Decapsulation (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 the 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 PTI of the CP. In this example, we assume one active PTI with Instance-ID I-ID1. Assume the controller instantiates PTIs by signaling Replication segments, i.e., the Replication-ID of these Replication segments is <Root, Tree-ID, Instance-ID>. All Replication segments use the Tree-SID T-SID1 as the 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 PTI 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. The Replication segment at R1 replicates to R2. The Replication segment at R2 replicates to R6 and R7. Note that 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 a 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 the role of a Leaf as well as a transit node replicating to R6 and R7. Replication to R6, using N-SID6, steers a packet via IGP shortest path to that node. Replication to R7, using N-SID7, steers a 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 CP 1 of the SR P2MP Policy at R1:

- Since R1 is directly connected to R2, R1 performs the PUSH operation with just the <T-SID1> label for the replicated copy and sends it to R2 on interface L12.
- R2, as a Leaf, performs the NEXT operation, pops the T-SID1 label, and delivers the payload. For replication to R6, R2 performs a PUSH operation of N-SID6 to send the <N-SID6,T-SID1> label stack to R3. R3 is the penultimate hop for N-SID6; it performs PHP, 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 the <N-SID7,T-SID1> label stack to R4, which is one of the IGP ECMP next hops (R5 is other) towards R7. R4 is the penultimate hop for N-SID7; it performs PHP, which corresponds to the NEXT operation, and the packet is then sent to R7 with <T-SID1> in the label stack.
- R6, as a Leaf, performs the NEXT operation, pops the T-SID1 label, and delivers the payload.
- R7, as a Leaf, performs the NEXT operation, pops the 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 an SR Policy, Policy27. The policy has one SID in the 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 a 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 the role of a Leaf as well as a transit node replicating to R6 and R7. Replication to R6 steers a packet via IGP shortest path to that node. Replication to R7, via an SR Policy, first encapsulates the packet using H.Encaps and then steers the outer packet to R4. End.X USD on R4 decapsulates the 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 CP 1 of the SR P2MP Policy at R1 using H.Encaps.Replicate behavior:

- Since R1 is directly connected to R2, R1 sends the replicated copy (2001:db8::1, 2001:db8:cccc:2:fa::) (A,B2) to R2 on interface L12.
- R2, as a Leaf, removes the 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 the 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 the outer IPv6 header, and sends (2001:db8::1, 2001:db8:cccc:7:fa::) (A,B2) to R7.
- R6, as a Leaf, removes the outer IPv6 header and delivers the payload.
- R7, as a Leaf, removes the outer IPv6 header and delivers the payload.

A.2. P2MP Tree with Adjacent Replication Segments

Assume the controller computes a PTI with Root node R1, Intermediate and Leaf node R2, Intermediate nodes R3 and R5, and Leaf nodes R6 and R7. The controller instantiates the PTI by stitching Replication segments at R1, R2, R3, R5, R6, and R7. The Replication segment at R1 replicates to R2. The Replication segment at R2 replicates to R3 and R5. The Replication segment at R3 replicates to R6. The Replication segment at R5 replicates to R7. Note that 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 a 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 the role of a Leaf as well as a transit node replicating to R3 and R5. Replication to R3 steers a packet directly to the node on L23. Replication to R5 steers a 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 a 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 a 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 the PUSH operation with just the <T-SID1> label for the replicated copy and sends it to R2 on interface L12.
- R2, as a Leaf, performs the NEXT operation, pops the T-SID1 label, and delivers the payload. It also performs the PUSH operation on T-SID1 for replication to R3 and R5. For replication to R6, R2 sends the <T-SID1> label stack to R3 on interface L23. For replication to R5, R2 sends the <T-SID1> label stack to R5 on interface L25.
- R3 performs the NEXT operation on T-SID1, performs a PUSH operation for replication to R6, and sends the <T-SID1> label stack to R6 on interface L36.
- R5 performs the NEXT operation on T-SID1, performs a PUSH operation for replication to R7, and sends the <T-SID1> label stack to R7 on interface L57.
- R6, as a Leaf, performs the NEXT operation, pops the T-SID1 label, and delivers the payload.
- R7, as a Leaf, performs the NEXT operation, pops the 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 a 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 the role of a Leaf as well as a transit node replicating to R3 and R5. Replication to R3 steers a packet directly to the node on L23. Replication to R5 steers a 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 a 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 a 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 CP 1 of the SR P2MP Policy at R1 using the H.Encaps.Replicate behavior:

- Since R1 is directly connected to R2, R1 sends the replicated copy (2001:db8::1, 2001:db8:cccc:2:fa::) (A,B2) to R2 on interface L12.
- R2, as a Leaf, removes the 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 a Leaf, removes the outer IPv6 header and delivers the payload.
- R7, as a Leaf, removes the outer IPv6 header and delivers the payload.

Acknowledgements

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

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,
United States of America
Email: arvvenka@cisco.com

Zafar Ali
Cisco Systems, Inc.
United States of America
Email: zali@cisco.com

Swadesh Agrawal
Cisco Systems, Inc.
San Jose,
United States of America
Email: swaagraw@cisco.com

Jayant Kotalwar
Nokia
Mountain View,
United States of America
Email: jayant.kotalwar@nokia.com

Tanmoy Kundu
Nokia
Mountain View,
United States of America
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.
United States of America
Email: abudhira@cisco.com

Mankamana Mishra
Cisco Systems, Inc.
United States of America
Email: mankamis@cisco.com

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