SPRING Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2021

C. Filsfils
K. Talaulikar, Ed.
Cisco Systems, Inc.
D. Voyer
Bell Canada
A. Bogdanov
Google, Inc.
P. Mattes
Microsoft

November 1, 2020

Segment Routing Policy Architecture
draft-ietf-spring-segment-routing-policy-09

Abstract

Segment Routing (SR) allows a headend node to steer a packet flow along any path. Intermediate per-flow states are eliminated thanks to source routing. The headend node steers a flow into an SR Policy. The header of a packet steered in an SR Policy is augmented with an ordered list of segments associated with that SR Policy. This document details the concepts of SR Policy and steering into an SR Policy.

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 May 5, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
1.1. Requirements Language 3
2. SR Policy 4
2.1. Identification of an SR Policy 4
2.2. Candidate Path and Segment List 5
2.3. Protocol-Origin of a Candidate Path 6
2.4. Originator of a Candidate Path 6
2.5. Discriminator of a Candidate Path 7
2.6. Identification of a Candidate Path 7
2.7. Preference of a Candidate Path 8
2.8. Validity of a Candidate Path 8
2.9. Active Candidate Path 8
2.10. Validity of an SR Policy 9
2.11. Instantiation of an SR Policy in the Forwarding Plane . . 9
2.12. Priority of an SR Policy 10
2.13. Summary 10
3. Segment Routing Database 11
4. Segment Types 12
4.1. Explicit Null 16
5. Validity of a Candidate Path 16
5.1. Explicit Candidate Path 16
5.2. Dynamic Candidate Path 18
5.3. Composite Candidate Path 18
6. Binding SID 19
6.1. BSID of a candidate path 19
6.2. BSID of an SR Policy 19
6.3. Forwarding Plane 20
6.4. Non-SR usage of Binding SID 21
7. SR Policy State 21
8. Steering into an SR Policy 21
8.1. Validity of an SR Policy 22
8.2. Drop upon invalid SR Policy 22
8.3. Incoming Active SID is a BSID 22
8.4. Per-Destination Steering 23
8.5. Recursion on an on-demand dynamic BSID 24
8.6. Per-Flow Steering 25

8.7. Policy-based Routing 26
8.8. Optional Steering Modes for BGP Destinations 26
9. Protection 28
9.1. Leveraging TI-LFA local protection of the constituent IGP segments 29
9.2. Using an SR Policy to locally protect a link 29
9.3. Using a Candidate Path for Path Protection 29
10. Security Considerations 30
11. IANA Considerations 30
11.1. Guidance for Designated Experts 31
12. Acknowledgement 31
13. Contributors 32
14. References 33
14.1. Normative References 33
14.2. Informative References 33
Authors' Addresses 36

1. Introduction

Segment Routing (SR) allows a headend node to steer a packet flow along any path. Intermediate per-flow states are eliminated thanks to source routing [RFC8402].

The headend node is said to steer a flow into an Segment Routing Policy (SR Policy).

The header of a packet steered into an SR Policy is augmented with an ordered list of segments associated with that SR Policy.

This document details the concepts of SR Policy and steering packets into an SR Policy. These apply equally to the MPLS and SRv6 instantiations of segment routing.

For reading simplicity, the illustrations are provided for the MPLS instantiations.

1.1. 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 Policy

An SR Policy is a framework that enables instantiation of an ordered list of segments on a node for implementing a source routing policy with a specific intent for traffic steering from that node.

The Segment Routing architecture [RFC8402] specifies that any instruction can be bound to a segment. Thus, an SR Policy can be built using any type of Segment Identifier (SID) including those associated with topological or service instructions.

This section defines the key aspects and constituents of an SR Policy.

2.1. Identification of an SR Policy

An SR Policy is identified through the tuple <headend, color, endpoint>. In the context of a specific headend, one may identify an SR policy by the <color, endpoint> tuple.

The headend is the node where the policy is instantiated/implemented. The headend is specified as an IPv4 or IPv6 address and is expected to be unique in the domain.

The endpoint indicates the destination of the policy. The endpoint is specified as an IPv4 or IPv6 address and is expected to be unique in the domain. In a specific case (refer to Section 8.8.1), the endpoint can be the null address (0.0.0.0 for IPv4, ::0 for IPv6).

The color is a 32-bit numerical value that associates the SR Policy with an intent (e.g. low-latency).

The endpoint and the color are used to automate the steering of service or transport routes on SR Policies (refer to Section 8).

An implementation MAY allow assignment of a symbolic name comprising of printable ASCII characters to an SR Policy to serve as a user-friendly attribute for debug and troubleshooting purposes. Such symbolic names may identify an SR Policy when the naming scheme ensures uniqueness. The SR Policy name may also be signaled along with a candidate path of the SR Policy (refer Section 2.2). An SR Policy may have multiple names associated with it in the scenario where the headend receives different SR Policy names along with candidate paths for the same SR Policy.

2.2. Candidate Path and Segment List

An SR Policy is associated with one or more candidate paths. A candidate path is the unit for signaling of an SR Policy to a headend via protocols like Path Computation Element (PCE) Communication Protocol (PCEP) [RFC8664] or BGP SR Policy [I-D.ietf-idr-segment-routing-te-policy].

A Segment-List represents a specific source-routed path to send traffic from the headend to the endpoint of the corresponding SR policy.

A candidate path is either dynamic, explicit or composite.

An explicit candidate path is expressed as a Segment-List or a set of Segment-Lists.

A dynamic candidate path expresses an optimization objective and a set of constraints. The headend (potentially with the help of a PCE) computes the solution Segment-List (or set of Segment-Lists) that solves the optimization problem.

If a candidate path is associated with a set of Segment-Lists, each Segment-List is associated with a weight for weighted load balancing (refer Section 2.11 for details). The default weight is 1.

A composite candidate path acts as a container for grouping of SR Policies. The composite candidate path construct enables combination of SR Policies, each with explicit candidate paths and/or dynamic candidate paths with potentially different optimization objectives and constraints, for a load-balanced steering of packet flows over its constituent SR Policies. The following criteria apply for inclusion of constituent SR Policies using a composite candidate path under a parent SR Policy:

- the endpoints of the constituent SR Policies and the parent SR Policy MUST be identical
- The colors of each of the constituent SR Policies and the parent SR Policy MUST be different
- the constituent SR Policies MUST NOT use composite candidate paths

Each constituent SR Policy of a composite candidate path is associated with a weight for load-balancing purposes (refer Section 2.11 for details). The default weight is 1.

2.3. Protocol-Origin of a Candidate Path

A headend may be informed about a candidate path for an SR Policy <color, endpoint> by various means including: via configuration, PCEP [RFC8664] or BGP [I-D.ietf-idr-segment-routing-te-policy].

Protocol-Origin of a candidate path is an 8-bit value which identifies the component or protocol that originates or signals the candidate path.

The head-end assigns different Protocol-Origin Priority values to each Protocol-Origin. The Protocol-Origin Priority is used as a tie-breaker between candidate-paths of equal preference, as described in Section 2.9. The table below specifies the RECOMMENDED default values of Protocol-Origin Priority:

Value	Protocol-Origin
10	PCEP
20	BGP SR Policy
30	Via Configuration

Table 1: Protocol-Origin Priority default values

Implementations MAY allow modifications of these default values assigned to protocols on the headend along similar lines as a routing administrative distance. Its application in the candidate path selection is described in Section 2.9.

2.4. Originator of a Candidate Path

Originator identifies the node which provisioned or signalled the candidate path on the headend. The originator is expressed in the form of a 160 bit numerical value formed by the concatenation of the fields of the tuple <ASN, node-address> as below:

- ASN : represented as a 4 byte number.
- Node Address : represented as a 128 bit value. IPv4 addresses are encoded in the lowest 32 bits.

Its application in the candidate path selection is described in Section 2.9.

When Protocol-Origin is Via Configuration, the ASN and node address MAY be set to either the headend or the provisioning controller/node ASN and address. Default value is 0 for both AS and node address.

When Protocol-Origin is PCEP, it is the IPv4 or IPv6 address of the PCE and the AS number SHOULD be set to 0 by default when not available or known.

When Protocol-Origin is BGP SR Policy, the ASN and Node Address are provided by BGP (refer [I-D.ietf-idr-segment-routing-te-policy]) on the headend.

2.5. Discriminator of a Candidate Path

The Discriminator is a 32 bit value associated with a candidate path that uniquely identifies it within the context of an SR Policy from a specific Protocol-Origin as specified below:

When Protocol-Origin is Via Configuration, this is an implementation's configuration model specific unique identifier for a candidate path. Default value is 0.

When PCEP is the Protocol-Origin, the method to uniquely identify signalled path will be specified in a future PCEP document. Default value is 0.

When BGP SR Policy is the Protocol-Origin, it is the distinguisher (refer Section 2.1 of [I-D.ietf-idr-segment-routing-te-policy]).

Its application in the candidate path selection is described in Section 2.9.

2.6. Identification of a Candidate Path

A candidate path is identified in the context of a single SR Policy.

A candidate path is not shared across SR Policies.

A candidate path is not identified by its Segment-List(s).

> If CP1 is a candidate path of SR Policy Pol1 and CP2 is a candidate path of SR Policy Pol2, then these two candidate paths are independent, even if they happen to have the same Segment-List. The Segment-List does not identify a candidate path. The Segment-List is an attribute of a candidate path.

The identity of a candidate path MUST be uniquely established in the context of an SR Policy <headend, color, endpoint> in order to handle

add, delete or modify operations on them in an unambiguous manner regardless of their source(s).

The tuple <Protocol-Origin, originator, discriminator> uniquely identifies a candidate path.

Candidate paths MAY also be assigned or signaled with a symbolic name comprising printable ASCII characters to serve as a user-friendly attribute for debug and troubleshooting purposes. Such symbolic names MUST NOT be considered as identifiers for a candidate path.

2.7. Preference of a Candidate Path

The preference of the candidate path is used to select the best candidate path for an SR Policy. The default preference is 100.

It is RECOMMENDED that each candidate path of a given SR policy has a different preference.

2.8. Validity of a Candidate Path

A candidate path is usable when it valid. A common path validity criterion is the reachability of its constituent SIDs. The validation rules are specified in Section 5.

2.9. Active Candidate Path

A candidate path is selected when it is valid and it is determined to be the best path of the SR Policy. The selected path is referred to as the "active path" of the SR policy in this document.

Whenever a new path is learned or an active path is deleted, the validity of an existing path changes or an existing path is changed, the selection process MUST be re-executed.

The candidate path selection process operates on the candidate path Preference. A candidate path is selected when it is valid and it has the highest preference value among all the candidate paths of the SR Policy.

In the case of multiple valid candidate paths of the same preference, the tie-breaking rules are evaluated on the identification tuple in the following order until only one valid best path is selected:

1. Higher value of Protocol-Origin Priority is selected.

2. If specified by configuration, prefer the existing installed path.

3. Lower value of originator is selected.

4. Finally, the higher value of discriminator is selected.

The rules are framed with multiple protocols and sources in mind and hence may not follow the logic of a single protocol (e.g. BGP best path selection). The motivation behind these rules are as follows:

- The Protocol-Origin allows an operator to setup a default selection mechanism across protocol sources, e.g., to prefer configured over paths signalled via BGP SR Policy or PCEP.

- The preference, being the first tiebreaker, allows an operator to influence selection across paths thus allowing provisioning of multiple path options, e.g., CP1 is preferred and if it becomes invalid then fall-back to CP2 and so on. Since preference works across protocol sources it also enables (where necessary) selective override of the default protocol-origin preference, e.g., to prefer a path signalled via BGP SR Policy over what is configured.

- The originator allows an operator to have multiple redundant controllers and still maintain a deterministic behaviour over which of them are preferred even if they are providing the same candidate paths for the same SR policies to the headend.

- The discriminator performs the final tiebreaking step to ensure a deterministic outcome of selection regardless of the order in which candidate paths are signalled across multiple transport channels or sessions.

[I-D.filsfils-spring-sr-policy-considerations] provides a set of examples to illustrate the active candidate path selection rules.

2.10. Validity of an SR Policy

An SR Policy is valid when it has at least one valid candidate path.

2.11. Instantiation of an SR Policy in the Forwarding Plane

A valid SR Policy is instantiated in the forwarding plane.

Only the active candidate path SHOULD be used for forwarding traffic that is being steered onto that policy.

If a set of Segment-Lists is associated with the active path of the policy, then the steering is per flow and W-ECMP based according to the relative weight of each Segment-List.

The fraction of the flows associated with a given Segment-List is w/ Sw where w is the weight of the Segment-List and Sw is the sum of the weights of the Segment-Lists of the selected path of the SR Policy.

When a composite candidate path is active, the fraction of flows steered into each constituent SR Policy is equal to the relative weight of each constituent SR Policy. Further load balancing of flows steered into a constituent SR Policy is performed based on the weights of the Segment-List of the active candidate path of that constituent SR Policy.

The accuracy of the weighted load-balancing depends on the platform implementation.

2.12. Priority of an SR Policy

Upon topological change, many policies could be recomputed or revalidated. An implementation MAY provide a per-policy priority configuration. The operator MAY set this field to indicate order in which the policies should be re-computed. Such a priority is represented by an integer in the range (0, 255) where the lowest value is the highest priority. The default value of priority is 128.

An SR Policy may comprise multiple Candidate Paths received from the same or different sources. A candidate path MAY be signaled with a priority value. When an SR Policy has multiple candidate paths with distinct signaled non-default priority values, the SR Policy as a whole takes the lowest value (i.e. the highest priority) amongst these signaled priority values.

2.13. Summary

In summary, the information model is the following:

```
SR policy POL1 <headend = H1, color = 1, endpoint = E1>
     Candidate-path CP1 <protocol-origin = 20, originator =
100:1.1.1.1, discriminator = 1>
          Preference 200
          Weight W1, SID-List1 <SID11...SID1i>
          Weight W2, SID-List2 <SID21...SID2j>
     Candidate-path CP2 <protocol-origin = 20, originator =
100:2.2.2.2, discriminator = 2>
          Preference 100
          Weight W3, SID-List3 <SID31...SID3i>
          Weight W4, SID-List4 <SID41...SID4j>
```

The SR Policy POL1 is identified by the tuple <headend, color, endpoint>. It has two candidate paths CP1 and CP2. Each is

identified by a tuple <protocol-origin, originator, discriminator>. CP1 is the active candidate path (it is valid and it has the highest preference). The two Segment-Lists of CP1 are installed as the forwarding instantiation of SR policy POL1. Traffic steered on POL1 is flow-based hashed on Segment-List <SID11...SID1i> with a ratio W1/(W1+W2).

The information model of SR Policy POL100 having a composite candidate path is the following:

```
SR policy POL100 <headend = H1, color = 100, endpoint = E1>
     Candidate-path CP1 <protocol-origin = 20, originator =
100:1.1.1.1, discriminator = 1>
         Preference 200
         Weight W1, SR policy <color = 1>
         Weight W2, SR policy <color = 2>
```

The constituent SR Policies POL1 and POL2 have information model as described at the start of this section. They are referenced only by color in the composite candidate path since their headend and endpoint are identical to the POL100. The valid Segment-Lists of the active candidate path of POL1 and POL2 are installed in the forwarding. Traffic steered on POL100 is flow-based hashed on POL1 with a ratio W1/(W1+W2). Within the POL1, the flow-based hashing over its Segment-Lists are performed as described earlier in this section.

3. Segment Routing Database

An SR headend maintains the Segment Routing Database (SR-DB). The SR-DB is a conceptual database to illustrate the various pieces of information and their sources that may help in SR Policy computation and validation. There is no specific requirement for an implementation to create a new database as such.

An SR headend leverages the SR-DB to validate explicit candidate paths and compute dynamic candidate paths.

The information in the SR-DB MAY include:

- IGP information (topology, IGP metrics based on ISIS [RFC1195] and OSPF [RFC2328] [RFC5340])
- Segment Routing information (such as SRGB, SRLB, Prefix-SIDs, Adj-SIDs, BGP Peering SID, SRv6 SIDs) [RFC8402] [I-D.ietf-spring-srv6-network-programming]
- TE Link Attributes (such as TE metric, SRLG, attribute-flag, extended admin group) [RFC5305] [RFC3630].

- Extended TE Link attributes (such as latency, loss) [RFC8570] [RFC7471]
- Inter-AS Topology information [I-D.ietf-idr-bgpls-segment-routing-epe].

The attached domain topology MAY be learned via IGP, BGP-LS or NETCONF.

A non-attached (remote) domain topology MAY be learned via BGP-LS or NETCONF.

In some use-cases, the SR-DB may only contain the attached domain topology while in others, the SR-DB may contain the topology of multiple domains and in this case it is multi-domain capable.

The SR-DB MAY also contain the SR Policies instantiated in the network. This can be collected via BGP-LS [I-D.ietf-idr-te-lsp-distribution] or PCEP [RFC8231] and [I-D.ietf-pce-binding-label-sid]. This information allows to build an end-to-end policy on the basis of intermediate SR policies (see Section 6 for further details).

The SR-DB MAY also contain the Maximum SID Depth (MSD) capability of nodes in the topology. This can be collected via ISIS [RFC8491], OSPF [RFC8476], BGP-LS [RFC8814] or PCEP [RFC8664].

The use of the SR-DB for computation and validation of SR Policies is outside the scope of this document. Some implementation aspects related to this are covered in [I-D.filsfils-spring-sr-policy-considerations].

4. Segment Types

A Segment-List is an ordered set of segments represented as <S1, S2, ... Sn> where S1 is the first segment.

Based on the desired dataplane, either the MPLS label stack or the SRv6 SRH is built from the Segment-List. However, the Segment-List itself can be specified using different segment-descriptor types and the following are currently defined:

Type A: SR-MPLS Label:
: A MPLS label corresponding to any of the segment types defined for SR-MPLS (as defined in [RFC8402] or other SR-MPLS specifications) can be used. Additionally, reserved labels like explicit-null or in general any MPLS label may also be used. E.g. this type can be used to specify a label representation which maps to an optical transport path on a

packet transport node. This type does not require the headend to perform SID resolution.

Type B: SRv6 SID:
An IPv6 address corresponding to any of the SID behaviors for SRv6 (as defined in [I-D.ietf-spring-srv6-network-programming] or other SRv6 specifications) can be used. This type does not require the headend to perform SID resolution. Optionally, the SRv6 SID behavior (as defined in [I-D.ietf-spring-srv6-network-programming] or other SRv6 specifications) and structure (as defined in [I-D.ietf-spring-srv6-network-programming]) MAY also be provided for the headend to perform validation of the SID when using it for building the Segment List.

Type C: IPv4 Prefix with optional SR Algorithm:
The headend is required to resolve the specified IPv4 Prefix Address to the SR-MPLS label corresponding to a Prefix SID segment (as defined in [RFC8402]). The SR algorithm (refer to Section 3.1.1 of [RFC8402]) to be used MAY also be provided.

Type D: IPv6 Global Prefix with optional SR Algorithm for SR-MPLS:
In this case the headend is required to resolve the specified IPv6 Global Prefix Address to the SR-MPLS label corresponding to its Prefix SID segment (as defined in [RFC8402]). The SR Algorithm (refer to Section 3.1.1 of [RFC8402]) to be used MAY also be provided.

Type E: IPv4 Prefix with Local Interface ID:
This type allows identification of Adjacency SID or BGP Peer Adjacency SID (as defined in [RFC8402]) label for point-to-point links including IP unnumbered links. The headend is required to resolve the specified IPv4 Prefix Address to the Node originating it and then use the Local Interface ID to identify the point-to-point link whose adjacency is being referred to. The Local Interface ID link descriptor follows semantics as specified in [RFC7752]. This type can also be used to indicate indirection into a layer 2 interface (i.e. without IP address) like a representation of an optical transport path or a layer 2 Ethernet port or circuit at the specified node.

Type F: IPv4 Addresses for link endpoints as Local, Remote pair:
This type allows identification of Adjacency SID or BGP Peer Adjacency SID (as defined in [RFC8402]) label for links. The headend is required to resolve the specified IPv4 Local Address to the Node originating it and then use the IPv4 Remote Address to identify the link adjacency being referred to. The Local

and Remote Address pair link descriptors follows semantics as specified in [RFC7752].

Type G: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SR-MPLS:

This type allows identification of Adjacency SID or BGP Peer Adjacency SID (as defined in [RFC8402]) label for links including those with only Link Local IPv6 addresses. The headend is required to resolve the specified IPv6 Prefix Address to the Node originating it and then use the Local Interface ID to identify the point-to-point link whose adjacency is being referred to. For other than point-to-point links, additionally the specific adjacency over the link needs to be resolved using the Remote Prefix and Interface ID. The Local and Remote pair of Prefix and Interface ID link descriptor follows semantics as specified in [RFC7752]. This type can also be used to indicate indirection into a layer 2 interface (i.e. without IP address) like a representation of an optical transport path or a layer 2 Ethernet port or circuit at the specified node.

Type H: IPv6 Addresses for link endpoints as Local, Remote pair for SR-MPLS:

This type allows identification of Adjacency SID or BGP Peer Adjacency SID (as defined in [RFC8402]) label for links with Global IPv6 addresses. The headend is required to resolve the specified Local IPv6 Address to the Node originating it and then use the Remote IPv6 Address to identify the link adjacency being referred to. The Local and Remote Address pair link descriptors follows semantics as specified in [RFC7752].

Type I: IPv6 Global Prefix with optional SR Algorithm for SRv6:

The headend is required to resolve the specified IPv6 Global Prefix Address to an SRv6 SID corresponding to a Prefix SID segment (as defined in [RFC8402]), such as a SID associated with the End behavior (as defined in [I-D.ietf-spring-srv6-network-programming]), of the node which is originating the prefix. The SR Algorithm (refer to Section 3.1.1 of [RFC8402]), the SRv6 SID behavior (as defined in [I-D.ietf-spring-srv6-network-programming] or other SRv6 specifications) and structure (as defined in [I-D.ietf-spring-srv6-network-programming]) MAY also be provided.

Type J: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SRv6:

This type allows identification of an SRv6 SID corresponding to an Adjacency SID or BGP Peer Adjacency SID (as defined in

[RFC8402]), such as a SID associated with the End.X behavior (as defined in [I-D.ietf-spring-srv6-network-programming]), associated with link or adjacency with only Link Local IPv6 addresses. The headend is required to resolve the specified IPv6 Prefix Address to the Node originating it and then use the Local Interface ID to identify the point-to-point link whose adjacency is being referred to. For other than point-to-point links, additionally the specific adjacency needs to be resolved using the Remote Prefix and Interface ID. The Local and Remote pair of Prefix and Interface ID link descriptor follows semantics as specified in [RFC7752]. The SR Algorithm (refer to Section 3.1.1 of [RFC8402]), the SRv6 SID behavior (as defined in [I-D.ietf-spring-srv6-network-programming] or other SRv6 specifications) and structure (as defined in [I-D.ietf-spring-srv6-network-programming]) MAY also be provided.

Type K: IPv6 Addresses for link endpoints as Local, Remote pair for SRv6:

This type allows identification of an SRv6 SID corresponding to an Adjacency SID or BGP Peer Adjacency SID (as defined in [RFC8402]), such as a SID associated with the End.X behavior (as defined in [I-D.ietf-spring-srv6-network-programming]), associated with link or adjacency with Global IPv6 addresses. The headend is required to resolve the specified Local IPv6 Address to the Node originating it and then use the Remote IPv6 Address to identify the link adjacency being referred to. The Local and Remote Address pair link descriptors follows semantics as specified in [RFC7752]. The SR Algorithm (refer to Section 3.1.1 of [RFC8402]), the SRv6 SID behavior (as defined in [I-D.ietf-spring-srv6-network-programming] or other SRv6 specifications) and structure (as defined in [I-D.ietf-spring-srv6-network-programming]) MAY also be provided.

When the algorithm is not specified for the SID types above which optionally allow for it, the headend SHOULD use the Strict Shortest Path algorithm if available; otherwise it SHOULD use the default Shortest Path algorithm. The specification of algorithm enables the use of IGP Flex Algorithm [I-D.ietf-lsr-flex-algo] specific SIDs in SR Policy.

For SID types C-through-K, a SID value may also be optionally provided to the headend for verification purposes. Section 5.1. describes the resolution and verification of the SIDs and Segment Lists on the headend.

When building the MPLS label stack or the IPv6 Segment list from the Segment List, the node instantiating the policy MUST interpret the set of Segments as follows:

- The first Segment represents the topmost label or the first IPv6 segment. It identifies the active segment the traffic will be directed toward along the explicit SR path.
- The last Segment represents the bottommost label or the last IPv6 segment the traffic will be directed toward along the explicit SR path.

4.1. Explicit Null

A Type A SID may be any MPLS label, including reserved labels.

For example, assuming that the desired traffic-engineered path from a headend 1 to an endpoint 4 can be expressed by the Segment-List <16002, 16003, 16004> where 16002, 16003 and 16004 respectively refer to the IPv4 Prefix SIDs bound to node 2, 3 and 4, then IPv6 traffic can be traffic-engineered from nodes 1 to 4 via the previously described path using an SR Policy with Segment-List <16002, 16003, 16004, 2> where mpls label value of 2 represents the "IPv6 Explicit NULL Label".

The penultimate node before node 4 will pop 16004 and will forward the frame on its directly connected interface to node 4.

The endpoint receives the traffic with top label "2" which indicates that the payload is an IPv6 packet.

When steering unlabeled IPv6 BGP destination traffic using an SR policy composed of Segment-List(s) based on IPv4 SIDs, the Explicit Null Label Policy is processed as specified in [I-D.ietf-idr-segment-routing-te-policy]) Section 2.4.4. When an "IPv6 Explicit NULL label" is not present as the bottom label, the headend SHOULD automatically impose one. Refer to Section 8 for more details.

5. Validity of a Candidate Path

5.1. Explicit Candidate Path

An explicit candidate path is associated with a Segment-List or a set of Segment-Lists.

An explicit candidate path is provisioned by the operator directly or via a controller.

The computation/logic that leads to the choice of the Segment-List is external to the SR Policy headend. The SR Policy headend does not compute the Segment-List. The SR Policy headend only confirms its validity.

An explicit candidate path MAY consist of a single explicit Segment-List containing only an implicit-null label to indicate pop-and-forward behavior. The BSID is popped and the traffic is forwarded based on the inner label or an IP lookup in the case of unlabeled IP packets. Such an explicit path can serve as a fallback or path of last resort for traffic being steered into an SR Policy using its BSID (refer to Section 8.3).

A Segment-List of an explicit candidate path MUST be declared invalid when:

- It is empty.
- Its weight is 0.
- The headend is unable to perform path resolution for the first SID into one or more outgoing interface(s) and next-hop(s).
- The headend is unable to perform SID resolution for any non-first SID of type C-through-K into an MPLS label or an SRv6 SID.
- The headend verification fails for any SID for which verification has been explicitly requested.

"Unable to perform path resolution" means that the headend has no path to the SID in its SR database.

SID verification is performed when the headend is explicitly requested to verify SID(s) by the controller via the signaling protocol used. Implementations MAY provide a local configuration option to enable verification on a global or per policy or per candidate path basis.

"Verification fails" for a SID means any of the following:

- The headend is unable to find the SID in its SR DB
- The headend detects mis-match between the SID value and its context provided for SIDs of type C-through-L in its SR DB.
- The headend is unable to perform SID resolution for any non-first SID of type C-through-K into an MPLS label or an SRv6 SID.

In multi-domain deployments, it is expected that the headend be unable to verify the reachability of the SIDs in remote domains. Types A or B MUST be used for the SIDs for which the reachability cannot be verified. Note that the first SID MUST always be reachable regardless of its type.

In addition, a Segment-List MAY be declared invalid when:

- Its last segment is not a Prefix SID (including BGP Peer Node-SID) advertised by the node specified as the endpoint of the corresponding SR policy.
- Its last segment is not an Adjacency SID (including BGP Peer Adjacency SID) of any of the links present on neighbor nodes and that terminate on the node specified as the endpoint of the corresponding SR policy.

An explicit candidate path is invalid as soon as it has no valid Segment-List.

5.2. Dynamic Candidate Path

A dynamic candidate path is specified as an optimization objective and constraints.

The headend of the policy leverages its SR database to compute a Segment-List ("solution Segment-List") that solves this optimization problem.

The headend re-computes the solution Segment-List any time the inputs to the problem change (e.g., topology changes).

When local computation is not possible (e.g., a policy's tailend is outside the topology known to the headend) or not desired, the headend MAY send path computation request to a PCE supporting PCEP extension specified in [RFC8664].

If no solution is found to the optimization objective and constraints, then the dynamic candidate path MUST be declared invalid.

[I-D.filsfils-spring-sr-policy-considerations] discusses some of the optimization objectives and constraints that may be considered by a dynamic candidate path. It illustrates some of the desirable properties of the computation of the solution Segment-List.

5.3. Composite Candidate Path

A composite candidate path is specified as a group of its constituent SR Policies.

A composite candidate path is valid when it has at least one valid constituent SR Policy.

6. Binding SID

The Binding SID (BSID) is fundamental to Segment Routing [RFC8402]. It provides scaling, network opacity and service independence. [I-D.filsfils-spring-sr-policy-considerations] illustrates some of these benefits. This section describes the association of BSID with an SR Policy.

6.1. BSID of a candidate path

Each candidate path MAY be defined with a BSID.

Candidate Paths of the same SR policy SHOULD have the same BSID.

Candidate Paths of different SR policies MUST NOT have the same BSID.

6.2. BSID of an SR Policy

The BSID of an SR Policy is the BSID of its active candidate path.

When the active candidate path has a specified BSID, the SR Policy uses that BSID if this value (label in MPLS, IPv6 address in SRv6) is available (i.e., not associated with any other usage: e.g. to another MPLS client, to another SID, to another SR Policy). In the case of SR-MPLS, an SRv6 BSID (e.g. with the behavior End.BM [I-D.ietf-spring-srv6-network-programming]) MAY be associated with the SR Policy in addition to the MPLS BSID. In the case of SRv6, multiple SRv6 BSIDs (e.g. with different behaviors like End.B6.Encap and End.B6.Encap.Red [I-D.ietf-spring-srv6-network-programming]) MAY be associated with the SR Policy.

Optionally, instead of only checking that the BSID of the active path is available, a headend MAY check that it is available within a given SID range i.e., Segment Routing Local Block (SRLB) as specified in [RFC8402].

When the specified BSID is not available (optionally is not in the SRLB), an alert message MUST be generated.

In the cases (as described above) where SR Policy does not have a BSID available, then the SR Policy MAY dynamically bind a BSID to itself. Dynamically bound BSID SHOULD use an available SID outside the SRLB.

Assuming that at time t the BSID of the SR Policy is B1, if at time t+dt a different candidate path becomes active and this new active path does not have a specified BSID or its BSID is specified but is

not available (e.g. it is in use by something else), then the SR Policy keeps the previous BSID B1.

The association of an SR Policy with a BSID thus MAY change over the life of the SR Policy (e.g., upon active path change). Hence, the BSID SHOULD NOT be used as an identification of an SR Policy.

6.2.1. Frequent use-case : unspecified BSID

All the candidate paths of the same SR Policy can have an unspecified BSID.

In such a case, a BSID MAY be dynamically bound to the SR Policy as soon as the first valid candidate path is received. That BSID is kept along all the life of the SR Policy and across changes of active candidate path.

6.2.2. Frequent use-case: all specified to the same BSID

All the paths of the SR Policy can have the same specified BSID.

6.2.3. Specified-BSID-only

An implementation MAY support the configuration of the Specified-BSID-only restrictive behavior on the headend for all SR Policies or individual SR Policies. Further, this restrictive behavior MAY also be signaled on a per SR Policy basis to the headend.

When this restrictive behavior is enabled, if the candidate path has an unspecified BSID or if the specified BSID is not available when the candidate path becomes active then no BSID is bound to it and it is considered invalid. An alert MUST be triggered for this error. Other candidate paths MUST then be evaluated for becoming the active candidate path.

6.3. Forwarding Plane

A valid SR Policy installs a BSID-keyed entry in the forwarding plane with the action of steering the packets matching this entry to the selected path of the SR Policy.

If the Specified-BSID-only restrictive behavior is enabled and the BSID of the active path is not available (optionally not in the SRLB), then the SR Policy does not install any entry indexed by a BSID in the forwarding plane.

6.4. Non-SR usage of Binding SID

An implementation MAY choose to associate a Binding SID with any type of interface (e.g. a layer 3 termination of an Optical Circuit) or a tunnel (e.g. IP tunnel, GRE tunnel, IP/UDP tunnel, MPLS RSVP-TE tunnel, etc). This enables the use of other non-SR enabled interfaces and tunnels as segments in an SR Policy Segment-List without the need of forming routing protocol adjacencies over them.

The details of this kind of usage are beyond the scope of this document. A specific packet optical integration use case is described in [I-D.anand-spring-poi-sr].

7. SR Policy State

The SR Policy State is maintained on the headend to represent the state of the policy and its candidate paths. This is to provide an accurate representation of whether the SR Policy is being instantiated in the forwarding plane and which of its candidate paths and segment-list(s) are active. The SR Policy state MUST also reflect the reason when a policy and/or its candidate path is not active due to validation errors or not being preferred.

The SR Policy state can be reported by the headend node via BGP-LS [I-D.ietf-idr-te-lsp-distribution] or PCEP [RFC8231] and [I-D.ietf-pce-binding-label-sid].

SR Policy state on the headend also includes traffic accounting information for the flows being steered via the policies. The details of the SR Policy accounting are beyond the scope of this document. The aspects related to the SR traffic counters and their usage in the broader context of traffic accounting in a SR network are covered in [I-D.filsfils-spring-sr-traffic-counters] and [I-D.ali-spring-sr-traffic-accounting] respectively.

Implementations MAY support an administrative state to control locally provisioned policies via mechanisms like CLI or NETCONF.

8. Steering into an SR Policy

A headend can steer a packet flow into a valid SR Policy in various ways:

- Incoming packets have an active SID matching a local BSID at the headend.
- Per-destination Steering: incoming packets match a BGP/Service route which recurses on an SR policy.

- Per-flow Steering: incoming packets match or recurse on a forwarding array of where some of the entries are SR Policies.
- Policy-based Steering: incoming packets match a routing policy which directs them on an SR policy.

For simplicity of illustration, this document uses the SR-MPLS example.

8.1. Validity of an SR Policy

An SR Policy is invalid when all its candidate paths are invalid as described in Section 5 and Section 2.10.

By default, upon transitioning to the invalid state,

- an SR Policy and its BSID are removed from the forwarding plane.
- any steering of a service (PW), destination (BGP-VPN), flow or packet on the related SR policy is disabled and the related service, destination, flow or packet is routed per the classic forwarding table (e.g. longest-match to the destination or the recursing next-hop).

8.2. Drop upon invalid SR Policy

An SR Policy MAY be enabled for the Drop-Upon-Invalid behavior:

- an invalid SR Policy and its BSID is kept in the forwarding plane with an action to drop.
- any steering of a service (PW), destination (BGP-VPN), flow or packet on the related SR policy is maintained with the action to drop all of this traffic.

The drop-upon-invalid behavior has been deployed in use-cases where the operator wants some PW to only be transported on a path with specific constraints. When these constraints are no longer met, the operator wants the PW traffic to be dropped. Specifically, the operator does not want the PW to be routed according to the IGP shortest-path to the PW endpoint.

8.3. Incoming Active SID is a BSID

Let us assume that headend H has a valid SR Policy P of Segment-List <S1, S2, S3> and BSID B.

When H receives a packet K with label stack <B, L2, L3>, H pops B and pushes <S1, S2, S3> and forwards the resulting packet according to SID S1.

"Forwarding the resulting packet according to S1" means: If S1 is an Adj SID or a PHP-enabled prefix SID advertised by a neighbor, H sends the resulting packet with label stack <S2, S3, L2, L3> on the outgoing interface associated with S1; Else H sends the resulting packet with label stack <S1, S2, S3, L2, L3> along the path of S1.

H has steered the packet into the SR policy P.

H did not have to classify the packet. The classification was done by a node upstream of H (e.g., the source of the packet or an intermediate ingress edge node of the SR domain) and the result of this classification was efficiently encoded in the packet header as a BSID.

This is another key benefit of the segment routing in general and the binding SID in particular: the ability to encode a classification and the resulting steering in the packet header to better scale and simplify intermediate aggregation nodes.

If the SR Policy P is invalid, the BSID B is not in the forwarding plane and hence the packet K is dropped by H.

8.4. Per-Destination Steering

Let us assume that headend H:

- learns a BGP route R/r via next-hop N, extended-color community C and VPN label V.
- has a valid SR Policy P to (color = C, endpoint = N) of Segment-List <S1, S2, S3> and BSID B.
- has a BGP policy which matches on the extended-color community C and allows its usage as SLA steering information.

If all these conditions are met, H installs R/r in RIB/FIB with next-hop = SR Policy P of BSID B instead of via N.

Indeed, H's local BGP policy and the received BGP route indicate that the headend should associate R/r with an SR Policy path to endpoint N with the SLA associated with color C. The headend therefore installs the BGP route on that policy.

This can be implemented by using the BSID as a generalized next-hop and installing the BGP route on that generalized next-hop.

When H receives a packet K with a destination matching R/r, H pushes the label stack <S1, S2, S3, V> and sends the resulting packet along the path to S1.

Note that any SID associated with the BGP route is inserted after the Segment-List of the SR Policy (i.e., <S1, S2, S3, V>).

The same behavior is applicable to any type of service route: any AFI/SAFI of BGP [RFC4760] any AFI/SAFI of LISP [RFC6830].

In a BGP multi-path scenario, the BGP route may be resolved over a mix of paths that include those that are steered over SR Policies and others resolved via the normal BGP nexthop resolution. Implementations MAY provide options to prefer one type over the other or other forms of local policy to determine the paths that are selected.

8.4.1. Multiple Colors

When a BGP route has multiple extended-color communities each with a valid SR Policy, the BGP process installs the route on the SR Policy giving preference to the color with the highest numerical value.

Let us assume that headend H:

- learns a BGP route R/r via next-hop N, extended-color communities C1 and C2 and VPN label V.
- has a valid SR Policy P1 to (color = C1, endpoint = N) of Segment-List <S1, S2, S3> and BSID B1.
- has a valid SR Policy P2 to (color = C2, endpoint = N) of Segment-List <S4, S5, S6> and BSID B2.
- has a BGP policy which matches on the extended-color communities C1 and C2 and allows their usage as SLA steering information

If all these conditions are met, H installs R/r in RIB/FIB with next-hop = SR Policy P2 of BSID=B2 (instead of N) because C2 > C1.

When the SR Policy with a specific color is not instantiated or in the down/inactive state, the SR Policy with the next highest numerical value of color is considered.

8.5. Recursion on an on-demand dynamic BSID

In the previous section, it was assumed that H had a pre-established "explicit" SR Policy (color C, endpoint N).

In this section, independently to the a-priori existence of any explicit candidate path of the SR policy (C, N), it is to be noted that the BGP process at headend node H triggers the instantiation of a dynamic candidate path for the SR policy (C, N) as soon as:

- the BGP process learns of a route R/r via N and with color C.

- a local policy at node H authorizes the on-demand SR Policy path instantiation and maps the color to a dynamic SR Policy path optimization template.

8.5.1. Multiple Colors

When a BGP route R/r via N has multiple extended-color communities Ci (with i=1 ... n), an individual on-demand SR Policy dynamic path request (color Ci, endpoint N) is triggered for each color Ci. The SR Policy that is used for steering is then determined as described in Section 8.4.1.

8.6. Per-Flow Steering

Let us assume that headend H:

- has a valid SR Policy P1 to (color = C1, endpoint = N) of Segment-List <S1, S2, S3> and BSID B1.
- has a valid SR Policy P2 to (color = C2, endpoint = N) of Segment-List <S4, S5, S6> and BSID B2.
- is configured to instantiate an array of paths to N where the entry 0 is the IGP path to N, color C1 is the first entry and Color C2 is the second entry. The index into the array is called a Forwarding Class (FC). The index can have values 0 to 7.
- is configured to match flows in its ingress interfaces (upon any field such as Ethernet destination/source/vlan/tos or IP destination/source/DSCP or transport ports etc.) and color them with an internal per-packet forwarding-class variable (0, 1 or 2 in this example).

If all these conditions are met, H installs in RIB/FIB:

- N via a recursion on an array A (instead of the immediate outgoing link associated with the IGP shortest-path to N).
- Entry A(0) set to the immediate outgoing link of the IGP shortest-path to N.
- Entry A(1) set to SR Policy P1 of BSID=B1.
- Entry A(2) set to SR Policy P2 of BSID=B2.

H receives three packets K, K1 and K2 on its incoming interface. These three packets either longest-match on N or more likely on a BGP/service route which recurses on N. H colors these 3 packets respectively with forwarding-class 0, 1 and 2. As a result:

- H forwards K along the shortest-path to N (which in SR-MPLS results in the pushing of the prefix-SID of N).
- H pushes <S1, S2, S3> on packet K1 and forwards the resulting frame along the shortest-path to S1.

- H pushes <S4, S5, S6> on packet K2 and forwards the resulting frame along the shortest-path to S4.

If the local configuration does not specify any explicit forwarding information for an entry of the array, then this entry is filled with the same information as entry 0 (i.e. the IGP shortest-path).

If the SR Policy mapped to an entry of the array becomes invalid, then this entry is filled with the same information as entry 0. When all the array entries have the same information as entry0, the forwarding entry for N is updated to bypass the array and point directly to its outgoing interface and next-hop.

The array index values (e.g. 0, 1 and 2) and the notion of forwarding-class are implementation specific and only meant to describe the desired behavior. The same can be realized by other mechanisms.

This realizes per-flow steering: different flows bound to the same BGP endpoint are steered on different IGP or SR Policy paths.

A headend MAY support options to apply per-flow steering only for traffic matching specific prefixes (e.g. specific IGP or BGP prefixes).

8.7. Policy-based Routing

Finally, headend H may be configured with a local routing policy which overrides any BGP/IGP path and steer a specified packet on an SR Policy. This includes the use of mechanisms like IGP Shortcut for automatic routing of IGP prefixes over SR Policies intended for such purpose.

8.8. Optional Steering Modes for BGP Destinations

8.8.1. Color-Only BGP Destination Steering

In the previous section, it is seen that the steering on an SR Policy is governed by the matching of the BGP route's next-hop N and the authorized color C with an SR Policy defined by the tuple (N, C).

This is the most likely form of BGP destination steering and the one recommended for most use-cases.

This section defines an alternative steering mechanism based only on the color.

This color-only steering variation is governed by two new "CO" bits defined in the color extended community in section 3 of [I-D.ietf-idr-segment-routing-te-policy].

The Color-Only flags "CO" are set to 00 by default.

When 00, the BGP destination is steered as follows:

```
    IF there is a valid SR Policy (N, C) where N is the IPv4 or IPv6

            endpoint address and C is a color;
        Steer into SR Policy (N, C);
    ELSE;
        Steer on the IGP path to the next-hop N.
```

This is the classic case described in this document previously and what is recommended in most scenarios.

When 01, the BGP destination is steered as follows:

```
    IF there is a valid SR Policy (N, C) where N is the IPv4 or IPv6

            endpoint address and C is a color;
        Steer into SR Policy (N, C);
    ELSE IF there is a valid SR Policy (null endpoint, C) of the
            same address-family of N;
        Steer into SR Policy (null endpoint, C);
    ELSE IF there is any valid SR Policy
            (any address-family null endpoint, C);
        Steer into SR Policy (any null endpoint, C);
    ELSE;
        Steer on the IGP path to the next-hop N.
```

When 10, the BGP destination is steered as follows:

```
    IF there is a valid SR Policy (N, C) where N is an IPv4 or IPv6
            endpoint address and C is a color;
        Steer into SR Policy (N, C);
    ELSE IF there is a valid SR Policy (null endpoint, C)
            of the same address-family of N;
        Steer into SR Policy (null endpoint, C);
    ELSE IF there is any valid SR Policy
            (any address-family null endpoint, C);
        Steer into SR Policy (any null endpoint, C);
    ELSE IF there is any valid SR Policy (any endpoint, C)
            of the same address-family of N;
        Steer into SR Policy (any endpoint, C);
    ELSE IF there is any valid SR Policy
```

```
            (any address-family endpoint, C);
        Steer into SR Policy (any address-family endpoint, C);
    ELSE;
        Steer on the IGP path to the next-hop N.
```

The null endpoint is 0.0.0.0 for IPv4 and ::0 for IPv6 (all bits set to the 0 value).

The value 11 is reserved for future use and SHOULD NOT be used. Upon reception, an implementations MUST treat it like 00.

8.8.2. Multiple Colors and CO flags

The steering preference is first based on highest color value and then CO-dependent for the color. Assuming a Prefix via (NH, C1(CO=01), C2(CO=01)); C1>C2 The steering preference order is:

- SR policy (NH, C1).
- SR policy (null, C1).
- SR policy (NH, C2).
- SR policy (null, C2).
- IGP to NH.

8.8.3. Drop upon Invalid

This document defined earlier that when all the following conditions are met, H installs R/r in RIB/FIB with next-hop = SR Policy P of BSID B instead of via N.

- H learns a BGP route R/r via next-hop N, extended-color community C and VPN label V.
- H has a valid SR Policy P to (color = C, endpoint = N) of Segment-List <S1, S2, S3> and BSID B.
- H has a BGP policy which matches on the extended-color community C and allows its usage as SLA steering information.

This behavior is extended by noting that the BGP policy may require the BGP steering to always stay on the SR policy whatever its validity.

This is the "drop upon invalid" option described in Section 8.2 applied to BGP-based steering.

9. Protection

9.1. Leveraging TI-LFA local protection of the constituent IGP segments

In any topology, Topology-Independent Loop Free Alternate (TI-LFA) [I-D.ietf-rtgwg-segment-routing-ti-lfa] provides a 50msec local protection technique for IGP SIDs. The backup path is computed on a per IGP SID basis along the post-convergence path.

In a network that has deployed TI-LFA, an SR Policy built on the basis of TI-LFA protected IGP segments leverages the local protection of the constituent segments. Since TI-LFA protection is based on IGP computation, there are cases where the path used during the fast-reroute time window may not meet the exact constraints of the SR Policy.

In a network that has deployed TI-LFA, an SR Policy instantiated only with non-protected Adj SIDs does not benefit from any local protection.

9.2. Using an SR Policy to locally protect a link

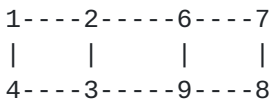


Figure 1: Local protection using SR Policy

An SR Policy can be instantiated at node 2 to protect the link 2to6. A typical explicit Segment-List would be <3, 9, 6>.

A typical use-case occurs for links outside an IGP domain: e.g. 1, 2, 3 and 4 are part of IGP/SR sub-domain 1 while 6, 7, 8 and 9 are part of IGP/SR sub-domain 2. In such a case, links 2to6 and 3to9 cannot benefit from TI-LFA automated local protection. The SR Policy with Segment-List <3, 9, 6> on node 2 can be locally configured to be a fast-reroute backup path for the link 2to6.

9.3. Using a Candidate Path for Path Protection

An SR Policy allows for multiple candidate paths, of which at any point in time there is a single active candidate path that is provisioned in the forwarding plane and used for traffic steering. However, another (lower preference) candidate path MAY be designated as the backup for a specific or all (active) candidate path(s). The following options are possible:

- A pair of disjoint candidate paths are provisioned with one of them as primary and the other is identified as its backup.
- A specific candidate path is provisioned as the backup for any (active) candidate path.
- The headend picks the next (lower) preference valid candidate path as the backup for the active candidate path.

The headend MAY compute a-priori and validate such backup candidate paths as well as provision them into forwarding plane as backup for the active path. A fast re-route mechanism MAY then be used to trigger sub 50msec switchover from the active to the backup candidate path in the forwarding plane. Mechanisms like BFD MAY be used for fast detection of such failures.

10. Security Considerations

This document specifies in detail the SR Policy construct introduced in [RFC8402] and its instantiation on a router supporting SR along with descriptions of mechanisms for steering of traffic flows over it. Therefore, the security considerations of [RFC8402] apply. This document does not define any new protocol extensions and does not introduce any further security considerations.

11. IANA Considerations

This document requests IANA to create a new top-level registry called "Segment Routing Parameters". This registry is being defined to serve as a top-level registry for keeping all other Segment Routing sub-registries.

The document also requests creation of a new sub-registry called "Segment Types" to be defined under the top-level "Segment Routing Parameters" registry. This sub-registry maintains the alphabetic identifiers for the segment types (as specified in section 4) that may be used within a Segment List of an SR Policy. This sub-registry would follow the Specification Required allocation policy as specified in [RFC8126].

The initial registrations for this sub-registry are as follows:

Value	Description	Reference
A	SR-MPLS Label	[This.ID]
B	SRv6 SID	[This.ID]
C	IPv4 Prefix with optional SR Algorithm	[This.ID]
D	IPv6 Global Prefix with optional SR Algorithm for SR-MPLS	[This.ID]
E	IPv4 Prefix with Local Interface ID	[This.ID]
F	IPv4 Addresses for link endpoints as Local, Remote pair	[This.ID]
G	IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SR-MPLS	[This.ID]
H	IPv6 Addresses for link endpoints as Local, Remote pair for SR-MPLS	[This.ID]
I	IPv6 Global Prefix with optional SR Algorithm for SRv6	[This.ID]
J	IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SRv6	[This.ID]
K	IPv6 Addresses for link endpoints as Local, Remote pair for SRv6	[This.ID]

Table 2: Initial IANA Registration

11.1. Guidance for Designated Experts

The Designated Expert (DE) is expected to ascertain the existence of suitable documentation (a specification) as described in [RFC8126] and to verify that the document is permanently and publicly available. The DE is also expected to check the clarity of purpose and use of the requested assignment. Additionally, the DE must verify that any request for one of these assignments has been made available for review and comment within the IETF: the DE will post the request to the SPRING Working Group mailing list (or a successor mailing list designated by the IESG). If the request comes from within the IETF, it should be documented in an Internet-Draft. Lastly, the DE must ensure that any other request for a code point does not conflict with work that is active or already published within the IETF.

12. Acknowledgement

The authors would like to thank Tarek Saad, Dhanendra Jain, Ruediger Geib, Rob Shakir, Cheng Li and Dhruv Dhody for their review comments and suggestions.

13. Contributors

The following people have contributed to this document:

Siva Sivabalan
Cisco Systems
Email: msiva@cisco.com

Zafar Ali
Cisco Systems
Email: zali@cisco.com

Jose Liste
Cisco Systems
Email: jliste@cisco.com

Francois Clad
Cisco Systems
Email: fclad@cisco.com

Kamran Raza
Cisco Systems
Email: skraza@cisco.com

Mike Koldychev
Cisco Systems
Email: mkoldych@cisco.com

Shraddha Hegde
Juniper Networks
Email: shraddha@juniper.net

Steven Lin
Google, Inc.
Email: stevenlin@google.com

Przemyslaw Krol
Google, Inc.
Email: pkrol@google.com

Martin Horneffer
Deutsche Telekom
Email: martin.horneffer@telekom.de

Dirk Steinberg
Steinberg Consulting
Email: dws@steinbergnet.net

Bruno Decraene
Orange Business Services
Email: bruno.decraene@orange.com

Stephane Litkowski
Orange Business Services
Email: stephane.litkowski@orange.com

Luay Jalil
Verizon
Email: luay.jalil@verizon.com

14. References

14.1. 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>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

14.2. Informative References

[I-D.ali-spring-sr-traffic-accounting] Filsfils, C., Talaulikar, K., Sivabalan, S., Horneffer, M., Raszuk, R., Litkowski, S., Voyer, D., and R. Morton, "Traffic Accounting in Segment Routing Networks", draft-ali-spring-sr-traffic-accounting-04 (work in progress), February 2020.

[I-D.anand-spring-poi-sr]
Anand, M., Bardhan, S., Subrahmaniam, R., Tantsura, J., Mukhopadhyaya, U., and C. Filsfils, "Packet-Optical Integration in Segment Routing", draft-anand-spring-poi-sr-08 (work in progress), July 2019.

[I-D.filsfils-spring-sr-policy-considerations]
Filsfils, C., Talaulikar, K., Krol, P., Horneffer, M., and P. Mattes, "SR Policy Implementation and Deployment Considerations", draft-filsfils-spring-sr-policy-considerations-06 (work in progress), October 2020.

[I-D.filsfils-spring-sr-traffic-counters]
Filsfils, C., Ali, Z., Horneffer, M., daniel.voyer@bell.ca, d., Durrani, M., and R. Raszuk, "Segment Routing Traffic Accounting Counters", draft-filsfils-spring-sr-traffic-counters-00 (work in progress), June 2018.

[I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgpls-segment-routing-epe-19 (work in progress), May 2019.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., Rosen, E., Jain, D., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-09 (work in progress), May 2020.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., Talaulikar, K., Dong, J., Chen, M., Gredler, H., and J. Tantsura, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-14 (work in progress), October 2020.

[I-D.ietf-lsr-flex-algo]
Psenak, P., Hegde, S., Filsfils, C., Talaulikar, K., and A. Gulko, "IGP Flexible Algorithm", draft-ietf-lsr-flex-algo-13 (work in progress), October 2020.

[I-D.ietf-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and C. Li, "Carrying Binding Label/Segment-ID in PCE-based Networks.", draft-ietf-pce-binding-label-sid-05 (work in progress), October 2020.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., Francois, P., Voyer, D., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-04 (work in progress), August 2020.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-24 (work in progress), October 2020.

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC3630] Katz, D., Kompella, K., and D. Yeung, "Traffic Engineering (TE) Extensions to OSPF Version 2", RFC 3630, DOI 10.17487/RFC3630, September 2003, <https://www.rfc-editor.org/info/rfc3630>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[RFC5305] Li, T. and H. Smit, "IS-IS Extensions for Traffic Engineering", RFC 5305, DOI 10.17487/RFC5305, October 2008, <https://www.rfc-editor.org/info/rfc5305>.

[RFC5340] Coltun, R., Ferguson, D., Moy, J., and A. Lindem, "OSPF for IPv6", RFC 5340, DOI 10.17487/RFC5340, July 2008, <https://www.rfc-editor.org/info/rfc5340>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <https://www.rfc-editor.org/info/rfc6830>.

[RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <https://www.rfc-editor.org/info/rfc7471>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC8231] Crabbe, E., Minei, I., Medved, J., and R. Varga, "Path Computation Element Communication Protocol (PCEP) Extensions for Stateful PCE", RFC 8231, DOI 10.17487/RFC8231, September 2017, <https://www.rfc-editor.org/info/rfc8231>.

[RFC8476] Tantsura, J., Chunduri, U., Aldrin, S., and P. Psenak, "Signaling Maximum SID Depth (MSD) Using OSPF", RFC 8476, DOI 10.17487/RFC8476, December 2018, <https://www.rfc-editor.org/info/rfc8476>.

[RFC8491] Tantsura, J., Chunduri, U., Aldrin, S., and L. Ginsberg, "Signaling Maximum SID Depth (MSD) Using IS-IS", RFC 8491, DOI 10.17487/RFC8491, November 2018, <https://www.rfc-editor.org/info/rfc8491>.

[RFC8570] Ginsberg, L., Ed., Previdi, S., Ed., Giacalone, S., Ward, D., Drake, J., and Q. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", RFC 8570, DOI 10.17487/RFC8570, March 2019, <https://www.rfc-editor.org/info/rfc8570>.

[RFC8664] Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "Path Computation Element Communication Protocol (PCEP) Extensions for Segment Routing", RFC 8664, DOI 10.17487/RFC8664, December 2019, <https://www.rfc-editor.org/info/rfc8664>.

[RFC8814] Tantsura, J., Chunduri, U., Talaulikar, K., Mirsky, G., and N. Triantafillis, "Signaling Maximum SID Depth (MSD) Using the Border Gateway Protocol - Link State", RFC 8814, DOI 10.17487/RFC8814, August 2020, <https://www.rfc-editor.org/info/rfc8814>.

Authors' Addresses

Clarence Filsfils
Cisco Systems, Inc.
Pegasus Parc
De kleetlaan 6a, DIEGEM BRABANT 1831
BELGIUM

Email: cfilsfil@cisco.com

Ketan Talaulikar (editor)
Cisco Systems, Inc.
India

Email: ketant@cisco.com

Daniel Voyer
Bell Canada
671 de la gauchetiere W
Montreal, Quebec H3B 2M8
Canada

Email: daniel.voyer@bell.ca

Alex Bogdanov
Google, Inc.

Email: bogdanov@google.com

Paul Mattes
Microsoft
One Microsoft Way
Redmond, WA 98052-6399
USA

Email: pamattes@microsoft.com