Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 1, 2018

C. Filsfils
S. Sivabalan
K. Raza
J. Liste
F. Clad
K. Talaulikar
Z. Ali
Cisco Systems, Inc.
S. Hegde
Juniper Networks, Inc.
D. Voyer
Bell Canada.
S. Lin
A. Bogdanov
P. Krol
Google, Inc.
M. Horneffer
Deutsche Telekom
D. Steinberg
Steinberg Consulting
B. Decraene
S. Litkowski
Orange Business Services
P. Mattes
Microsoft

February 28, 2018

Segment Routing Policy for Traffic Engineering
draft-filsfils-spring-segment-routing-policy-05.txt

Abstract

Segment Routing 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 the ordered list of segments associated with that SR Policy. This document details the concepts of SR Policy and steering into an SR Policy.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

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 [I-D.ietf-spring-segment-routing].

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

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

This document details the concepts of SR Policy and steering 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.

2. 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.

The endpoint indicates the destination of the policy. The endpoint is specified as an IPv4 or IPv6 address. 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).

2.2. Candidate Path and Segment List

An SR Policy is associated with one or more candidate paths.

A candidate path is itself associated with a Segment-List (SID-List) or a set of SID-Lists. In the latter case, each SID-List is associated with a weight for weighted load balancing (refer to section 2.11 for details). The default weight is 1.

A SID-List represents a specific source-routed way to send traffic from the head-end to the endpoint of the corresponding SR policy.

A candidate path is either dynamic or explicit.

An explicit candidate path is associated with a SID-List or a set of SID-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 SID-List (or set of SID-Lists) that solves the optimization problem.

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 [I-D.ietf-pce-pce-initiated-lsp] or BGP [I-D.draft-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 table below specifies the RECOMMENDED default values. Implementations MAY allow modifications of these default values assigned to protocols on the SRTE head-end as long as no two protocols share the same value.

The default values are listed below:

Value	Protocol-Origin
10	PCEP
20	BGP-SRTE
30	Local (via CLI, Yang model through NETCONF, gRPC, etc.)

Table 1: Protocol-origin Identifier

2.4. Originator of a Candidate Path

Originator identifies the node which provisioned or signalled the candidate path on the SRTE head-end. 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.

When Protocol-Origin is Local, the ASN and node address MAY be set to either the SRTE 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.

Protocol-Origin is BGP-SRTE, it is provided by the BGP component on the headend and is:

- the BGP Router ID and ASN of the node/controller signalling the candidate path when it has a BGP session to the headend, OR
- the BGP Router ID of the eBGP peer signalling the candidate path along with ASN of origin when the signalling is done via one or more intermediate eBGP routers, OR
- the BGP Originator ID [rfc4456] and the ASN of the node/controller when the signalling is done via one or more route-reflectors over iBGP session.

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 Local, this is an implementation's configuration model specific unique identifier for a candidate path.

When PCEP is the Protocol-Origin, the method to uniquely identify signalled path will be specified in an upcoming PCEP draft.

When BGP-SRTE is the Protocol-Origin, it is the distinguisher specified in Section 2.1 of [I.D.draft-ietf-idr-segment-routing-te-policy].

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 SID-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 SID-List. The SID-List does not identify a candidate path. The SID-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 identify 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 valid if it is usable. 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 is selected.

2. Lower value of originator is selected.

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

An implementation MAY choose to override any of the tie-breaking rules above and maintain the already selected candidate path as active path.

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 locally provisioned over paths signalled via BGP-SRTE 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-SRTE over what is locally provisioned.

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.

Section 14.3 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 is used for forwarding traffic that is being steered onto that policy.

If a set of SID-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 SID-List.

The fraction of the flows associated with a given SID-List is w/Sw where w is the weight of the SID-List and Sw is the sum of the weights of the SID-Lists of the selected path of the 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. An implementation MAY provide a per-policy priority field. 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.

2.13. Summary

In summary, the information model is the following:

```
         SR policy POL1 <headend, color, endpoint>
           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 SID-Lists of CP1 are installed as the forwarding instantiation of SR policy Pol1. Traffic steered on Pol1 is flow-based hashed on SID-List <SID11...SID1i> with a ratio W1/(W1+W2).

3. Segment Routing Database

An SR headend maintains the Segment Routing Traffic Engineering Database (SRTE-DB).

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

The information in the SRTE-DB MAY include:

- o IGP information (topology, IGP metrics).
- o TE Link Attributes (such as TE metric, SRLG, attribute-flag, extended admin group) [RFC5305, RFC3630].
- o Extended TE Link attributes (such as latency, loss) [RFC7810, RFC7471].
- o Inter-Domain Topology information [I.D.draft-ietf-idr-bgpls-segment-routing-epe].
- o Segment Routing information (such as SRGB, SRLB, Prefix-SIDs, Adj-SIDs, BGP Peering SID, SRv6 SIDs).

The SRTE-DB is multi-domain capable.

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 SRTE-DB may only contain the attached domain topology while in others, the SRTE-DB may contain the topology of multiple domains. The SRTE-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 ([I-D.ietf-pce-stateful-pce] and [I-D.sivabalan-pce-binding-label-sid]).

This information allows to build an end-to-end policy on the basis of intermediate SR policies (Section 6).

The SRTE-DB MAY also contain the Maximum SID Depth (MSD) capability of nodes in the topology. This can be collected via ISIS [draft-ietf-isis-segment-routing-msd], OSPF [draft-ietf-ospf-segment-routing-msd], BGP-LS [draft-ietf-idr-bgp-ls-segment-routing-msd] or PCEP [I-D.ietf-pce-segment-routing].

4. Segment Types

A SID-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 SID-List. However, the SID-List itself can specified using different segment-descriptor types and the following are defined:

Type 1: SR-MPLS Label:
SR-MPLS label corresponding to any of the segment types defined in [I.D.draft-ietf-spring-segment-routing] 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 SRTE process on the headend to perform any resolution.

Type 2: SRv6 SID:
IPv6 address corresponding to any of the segment types defined in [I.D.draft-filsfils-spring-srv6-network-programming] can be used. This type does not require the SRTE process on the headend to perform any resolution.

Type 3: IPv4 Prefix with optional SR Algorithm:
The SRTE process on the headend is required to resolve the specified IPv4 Prefix Address to the SR-MPLS label corresponding to its Prefix SID segment. The SR algorithm (refer to Section 3.1.1 of [I.D.draft-ietf-spring-segment-routing]) to be used MAY also be provided. When algorithm is not specified, the SRTE process is expected to use the Prefix SID signalled for the Strict Shortest Path algorithm when available and if not then use the Shortest Path or default algorithm.

Type 4: IPv6 Global Prefix with optional SR Algorithm for SR-MPLS:
In this case the SRTE process on the headend is required to resolve the specified IPv6 Global Prefix Address to the SR-MPLS label corresponding to its Prefix SID segment. The SR Algorithm (refer to Section 3.1.1 of [I.D.draft-ietf-spring-segment-routing]) to be used MAY also be provided. When algorithm is not specified, the SRTE process is expected to use the Prefix SID signalled for the Strict Shortest Path algorithm when available and if not then use the Shortest Path or default algorithm.

Type 5: IPv4 Prefix with Local Interface ID:
This type allows identification of Adjacency SID or BGP EPE Peer Adjacency SID label for point-to-point links including IP unnumbered links. The SRTE process on 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 6: IPv4 Addresses for link endpoints as Local, Remote pair:
This type allows identification of Adjacency SID for BGP EPE Peer Adjacency SID label for links. The SRTE process on 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 7: IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SR-MPLS:
This type allows identification of Adjacency SID or BGP EPE Peer Adjacency SID label for links including those with only Link Local IPv6 addresses. The SRTE process on 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 8: IPv6 Addresses for link endpoints as Local, Remote pair for SR-MPLS:
This type allows identification of Adjacency SID for BGP EPE Peer Adjacency SID label for links with Global IPv6 addresses. The SRTE process on 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 9: IPv6 Global Prefix with optional SR Algorithm for SRv6:
The SRTE process on the headend is required to resolve the specified IPv6 Global Prefix Address to the SRv6 END function SID corresponding to the node which is originating the prefix. The SR Algorithm (refer to Section 3.1.1 of [I.D.draft-ietf-spring-segment-routing]) to be used MAY also be provided. When algorithm is not specified, the SRTE process is expected to use the Prefix SID signaled for the Strict Shortest Path algorithm when available and if not then use the Shortest Path or default algorithm.

Type 10:IPv6 Prefix and Interface ID for link endpoints as Local, Remote pair for SRv6:
This type allows identification of SRv6 END.X SID for links with only Link Local IPv6 addresses. The SRTE process on 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.

Type 11:IPv6 Addresses for link endpoints as Local, Remote pair for SRv6:
This type allows identification of SRv6 END.X SID for links with Global IPv6 addresses. The SRTE process on 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.

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:

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

4.1. Explicit Null

A Type 1 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 SID-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 SID-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 SID-List(s) based on IPv4 SIDs, the Explicit Null Label Policy is processed as specified in draft-idr-segment-routing-te-policy Section 2.4.4. When this is not present then the headend SHOULD automatically impose the "IPv6 Explicit NULL Label" as bottom of stack label. Refer to "Steering" section later in this document.

5. Validity of a Candidate Path

5.1. Explicit Candidate Path

An explicit candidate path is associated with a SID-List or a set of SID-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 SID list is external to the SR Policy headend. The SR Policy headend does not compute the SID list. The SR Policy headend only confirms its validity.

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

- o It is empty.
- o Its weight is 0.
- o The headend is unable to resolve the first SID into one or more outgoing interface(s) and next-hop(s).
- o The headend is unable to resolve any non-first SID of type 3-to-11 into an MPLS label or an SRv6 SID.

"Unable to resolve" means that the headend has no path to the SID in its SRTE-DB.

In multi-domain deployments, it is expected that the headend be unable to verify the reachability of the SIDs in remote domains. Types 1 and 2 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 SID-List MAY be declared invalid when:

- o 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.
- o 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 SID-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 SRTE-DB to compute a SID-List ("solution SID-List") that solves this optimization problem.

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

When local computation is not possible (e.g., a policy's tail-end is outside the topology known to the head-end) or not desired, the head-end MAY send path computation request to a PCE supporting PCEP extension specified in [I-D.ietf-pce-segment-routing].

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

Section 14.4 lists 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 SID list.

6. Binding SID

The Binding SID (BSID) is fundamental to Segment Routing [I.D.draft-ietf-spring-segment-routing]. It provides scaling, network opacity and service independence. Section 14.5 illustrates these benefits.

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).

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., SRLB).

When the specified BSID is not available (optionally is not in the SRLB), an alert message is 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, then the SR Policy keeps the previous BSID B1.

6.2.1. Frequent use-cases : unspecified BSID

All the candidate paths of the same SR Policy have 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 path.

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

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

6.2.3. Specified-BSID-only

A headend MAY be configured with the Specified-BSID-only restrictive behavior.

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 is triggered. Other candidate paths can 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. Not an identification

The association of an SR Policy to a BSID MAY change over the life of the SR policy (e.g., upon active path change). The BSID of an SR Policy is not an identification of an SR policy. The identification of an SR Policy is the tuple <headend, color, endpoint>.

7. SR Policy State

The SR Policy State is maintained on the headend by the SRTE process represents the state of the policy and its candidate paths to provide the accurate representation of whether the policy is being instantiated in the forwarding plane and which of the candidate paths is 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.

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:

- o Incoming packets have an active SID matching a local BSID at the head-end.
- o Per-destination Steering: incoming packets match a BGP/Service route which recurses on an SR policy.
- o Per-flow Steering: incoming packets match or recurse on a forwarding array of where some of the entries are SR Policies.
- o 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.

By default, upon transitioning to the invalid state,

- o an SR Policy and its BSID are removed from the forwarding plane.
- o 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:

- o an invalid SR Policy and its BSID is kept in the forwarding plane with an action to drop.
- o 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 SID-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 in 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:

o learns a BGP route R/r via next-hop N, extended-color community C and VPN label V.
o has a valid SR Policy P to (endpoint = N, color = C) of SID-List <S1, S2, S3> and BSID B.
o has a BGP policy which matches on the extended-color community C and allows its usage as an SRTE 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 SRTE path to 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 SID-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 ([ID.draft-ietf-idr-tunnel-encaps-07], [I.D.draft-ietf-idr-segment-routing-te-policy]), any AFI/SAFI of LISP [RFC6830].

8.4.1. Multiple Colors

When a BGP route has multiple extended-color communities each with a valid SRTE policy, the BGP process installs the route on the SR policy whose color is of highest numerical value.

Let us assume that headend H:

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

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 (endpoint N, color C).

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

- o the BGP process learns of a route R/r via N and with color C.
- o a local policy at node H authorizes the on-demand SRTE path instantiation and maps the color to a dynamic SRTE 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 SRTE dynamic path request (endpoint N, color Ci) is triggered for each color Ci.

8.6. Per-Flow Steering

Let us assume that head-end H:

- o has a valid SR Policy P1 to (endpoint = N, color = C1) of SID-List <S1, S2, S3> and BSID B1.
- o has a valid SR Policy P2 to (endpoint = N, color = C2) of SID-List <S4, S5, S6> and BSID B2.
- o 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.

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

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 flags "C" and "O" defined in the color extended community [ref draft-ietf-idr-segment-routing-te-policy section 3].

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/v6
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 IPv4/6
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 IPv4/6
```

```
      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 (endpoint = N, color = C) of SID-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 an SRTE 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 10.2 applied to BGP-based steering.

9. Other type of SR Policies

9.1. Layer 2 and Optical Transport

```
                           1----2----3----4----5
                  I2(lambda L241)\       / I4(lambda L241)
                                  Optical
```

Figure 1: SR Policy with integrated DWDM

An explicit candidate path can express a path through a transport layer beneath IP (ATM, FR, DWDM). The transport layer could be ATM, FR, DWDM, back-to-back Ethernet etc. The transport path is modelled as a link between two IP nodes with the specific assumption that no distributed IP routing protocol runs over the link. The link may have IP address or be IP unnumbered. Depending on the transport protocol case, the link can be a physical DWDM interface and a lambda (integrated solution), an Ethernet interface and a VLAN, an ATM interface with a VPI/VCI, a FR interface with a DLCI etc.

Using the DWDM integrated use-case of Figure 1 as an illustration, let us assume

- o nodes 1, 2, 3, 4 and 5 are IP routers running an SR-enable IGP on the links 1-2, 2-3, 3-4 and 4-5.
- o The SRGB is homogeneous [16000, 24000].
- o Node K's prefix SID is 16000+K.
- o node 2 has an integrated DWDM interface I2 with Lambda L1.
- o node 4 has an integrated DWDM interface I4 with Lamdda L2.
- o the optical network is provisioned with a circuit from 2 to 4 with continuous lambda L241 (details outside the scope of this document).
- o Node 2 is provisioned with an SR policy with SID list <I2(L241)> and Binding SID B where I2(L241) is of type 5 (IPv4) or type 7 (IPv6), see section 4.
- o node 1 steers a packet P1 towards the prefix SID of node 5 (16005).
- o node 1 steers a packet P2 on the SR policy <16002, B, 16005>.

In such a case, the journey of P1 will be 1-2-3-4-5 while the journey of P2 will be 1-2-lambda(L241)-4-5. P2 skips the IP hop 3 and leverages the DWDM circuit from node 2 to node 4. P1 follows the shortest-path computed by the distributed routing protocol. The path of P1 is unaltered by the addition, modification or deletion of optical bypass circuits.

The salient point of this example is that the SRTE architecture seamlessly support explicit candidate paths through any transport sub-layer.

BGP-LS Extensions to describe the sub-IP-layer characteristics of the SR Policy are out of scope of this document (e.g. in Figure 1, the DWDM characteristics of the SR Policy at node 2 in terms of latency, loss, security, domain/country traversed by the circuit etc.).

9.2. Spray SR Policy

A Spray SRTE policy is a variant of an SRTE policy which involves packet replication.

Any traffic steered into a Spray SR Policy is replicated along the SID-Lists of its selected path.

In the context of a Spray SR Policy, the selected path SHOULD have more than one SID-List. The weights of the SID-Lists is not applicable for a Spray SR Policy. They MUST be set to 1.

Like any SR policy, a Spray SR Policy has a BSID instantiated into the forwarding plane.

Traffic is typically steered into a Spray SR Policy in two ways:

- o local policy-based routing at the headend of the policy.
- o remote classification and steering via the BSID of the Spray SR Policy.

10. 50msec Local Protection

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

In any topology, Topology-Independent LFA (TI-LFA) [I.D.draft-bashandy-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 leverage the local protection of the constituent segments.

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.

10.2. Using an SR Policy to locally protect a link

```
                        1----2-----6----7
                        |    |     |    |
                        4----3-----9----8
```

Figure 2: Local protection using SR Policy

An SR Policy can be instantiated at node 2 to protect the link 2to6. A typical explicit SID 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.

11. Other types of Segments

The Segment Routing architecture specifies that any instruction can be bound to a segment.

Similarly, an SR Policy can be composed of SIDs of any types.

On top of the classic IGP SIDs, BGP SIDs and BSIDs, this section highlights the use of service SIDs and IGP-Flex-Alg SIDs.

11.1. Service SID

A Service Segment is a Segment associated with a service, either directly or via an SR proxy. A service may be a physical appliance running on dedicated hardware, a virtualized service inside an isolated environment such as a VM, container or namespace, or any process running on a compute element [I.D.draft-clad-spring-segment-routing-service-chaining].

An SR Policy can be composed of a mix of segments of various types: IGP segments, BGP segments, Binding SIDs and Service Segments.

Similarly to other segments, service segments can be discovered via BGP-LS [I.D.draft-dawra-idr-bgp-sr-service-chaining].

11.2. Flex-Alg IGP SID

```
                  1--RED--2-------6
                  |       |       |
                  4-------3--RED--9
```

Figure 3: Illustration for Flex-Alg SID

Let us assume that

- o 1, 2, 3 and 4 are part of IGP 1.
- o 2, 6, 9 and 3 are part of IGP 2.
- o All the IGP link costs are 10.
- o Links 1to2 and 3to9 are colored with IGP Link Affinity Red.
- o Flex-Alg1 is defined in both IGPs as: avoid red, minimize IGP metric.
- o All nodes of each IGP domain are enabled for FlexAlg1
- o SID(k, 0) represents the PrefixSID of node k according to Alg=0.
- o SID(k, FlexAlg1) represents the PrefixSID of node k according to Flex-Alg1.

A controller can steer a flow from 1 to 9 through an end-to-end path that avoids the RED links of both IGP domains thanks to the explicit SR Policy <SID(2, FlexAlg1), SID9(FlexAlg1)>.

12. Binding SID to a tunnel

A Binding SID can be bound to any type of tunnel: IP tunnel, GRE tunnel, IP/UDP tunnel, MPLS RSVP-TE tunnel, etc.

13. Traffic Accounting

This section describes counters for traffic accounting in segment routing networks. The essence of Segment Routing consists in scaling the network by only maintaining per-flow state at the source or edge of the network. Specifically, only the headend of an SR policy maintains the related per-policy state. Egress and Midpoints along the source route do not maintain any per-policy state. The traffic counters described in this section respects the architecture principles of SR, while given visibility to the service provider for network operation and capacity planning. The traffic counters are divided into four categories: interface counters, prefix counters,

counters to measure the traffic (demand) matrix and SR policy counters at the policy head-end.

13.1. Traffic Counters Naming convention

The section uses the following naming convention when referring to the various counters. This is done in order to assign mnemonic names to SR counters.

- o The term counter(s) in all of the definitions specified in this document refers either to the (packet, byte) counters or the byte counter.
- o SR: any traffic whose FIB lookup is a segment (IGP prefix/Adj segments, BGP segments, any type of segments) or the matched FIB entry is steered on an SR Policy.
- o INT in name indicates a counter is implemented at a per interface level.
- o E in name refers to egress direction (with respect to the traffic flow).
- o I in name refers to ingress direction (with respect to the traffic flow).
- o TC in name indicates a counter is implemented on a Traffic Class (TC) basis.
- o TM in name refers to a Traffic Matrix (TM) counter.
- o PRO in name indicates that the counter is implemented on per protocol/adjacency type basis. Per PRO counters in this document can either be accounts for:

    * LAB (Labelled Traffic): the matched FIB entry is a segment, and the outgoing packet has at least one label (that label does not have to be a segment label, e.g., the label may be a VPN label).
    * V4 (IPv4 Traffic): the matched FIB entry is a segment which is PoP'ed. The outgoing packet is IPv4.
    * V6 (IPv6 Traffic): the matched FIB entry is a segment which is PoP'ed. The outgoing packet is IPv6.
- o POL in name refers to a Policy counter.
- o BSID in name indicates a policy counter for labelled traffic.
- o SL in name indicates a policy counter is implemented at a Segment-List (SL) level.

Counter nomenclature is exemplified using the following example:

- o SR.INT.E.PRO: Per-interface per-protocol aggregate egress SR traffic.
- o POL.BSID: Per-SR Policy labelled steered aggregate traffic counter.

13.2. Per-Interface SR Counters

For each local interface, node N maintains the following per-interface SR counters. These counters include accounting due to push, pop or swap operations on SR traffic.

13.2.1. Per interface, per protocol aggregate egress SR traffic counters (SR.INT.E.PRO)

The following counters are included under this category.

- o SR.INT.E.LAB: For each egress interface (INT.E), N MUST maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as labelled traffic.
- o SR.INT.E.V4: For each egress interface (INT.E), N MUST maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as IPv4 traffic (due to the pop operation).
- o SR.INT.E.V6: For each egress interface (INT.E), N MUST maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as IPv6 traffic (due to the pop operation).

13.2.2. Per interface, per traffic-class, per protocol aggregate egress SR traffic counters (SR.INT.E.PRO.TC)

This counter provides per Traffic Class (TC) breakdown of SR.INT.E.PRO. The following counters are included under this category.

- o SR.INT.E.LAB.TC: For each egress interface (INT.E) and a given Traffic Class (TC), N SHOULD maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as labelled traffic.
- o SR.INT.E.V4.TC: For each egress interface (INT.E) and a given Traffic Class (TC), N SHOULD maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as IPv4 traffic (due to the pop operation).
- o SR.INT.E.V6.TC: For each egress interface (INT.E) and a given Traffic Class (TC), N SHOULD maintain counter(s) for the aggregate SR traffic forwarded over the (INT.E) interface as IPv6 traffic (due to the pop operation).

13.2.3. Per interface aggregate ingress SR traffic counter (SR.INT.I)

The SR.INT.I counter is defined as follows:

For each ingress interface (INT.I), N SHOULD maintain counter(s) for the aggregate SR traffic received on I.

13.2.4. Per interface, per TC aggregate ingress SR traffic counter (SR.INT.I.TC)

This counter provides per Traffic Class (TC) breakdown of the SR.INT.I. It is defined as follow:

For each ingress interface (INT.I) and a given Traffic Class (TC), N MAY maintain counter(s) for the aggregate SR traffic (matching the traffic class TC criteria) received on I.

13.3. Prefix SID Counters

For a remote prefix SID S, node N maintains the following prefix SID counters. These counters include accounting due to push, pop or swap operations on the SR traffic.

13.3.1. Per-prefix SID egress traffic counter (PSID.E)

This counter is defined as follows:

For a remote prefix SID S, N MUST maintain counter(s) for aggregate traffic forwarded towards S.

13.3.2. Per-prefix SID per-TC egress traffic counter (PSID.E.TC)

This counter provides per Traffic Class (TC) breakdown of PSID.E. It is defined as follows:

For a given Traffic Class (TC) and a remote prefix SID S, N SHOULD maintain counter(s) for traffic forwarded towards S.

13.3.3. Per-prefix SID, per egress interface traffic counter (PSID.INT.E)

This counter is defined as follows:

For a given egress interface (INT.E) and a remote prefix SID S, N SHOULD maintain counter(s) for traffic forwarded towards S over the (INT.E) interface.

13.3.4. Per-prefix SID per TC per egress interface traffic counter (PSID.INT.E.TC)

This counter provides per Traffic Class (TC) breakdown of PSID.INT.E. It is defined as follows:

For a given Traffic Class (TC), an egress interface (INT.E) and a remote prefix SID S, N MAY maintain counter(s) for traffic forwarded towards S over the (INT.E) interface.

13.3.5. Per-prefix SID, per ingress interface traffic counter (PSID.INT.I)

This counter is defined as follows:

For a given ingress interface (INT.I) and a remote prefix SID S, N MAY maintain counter(s) for the traffic received on I and forwarded towards S.

13.3.6. Per-prefix SID, per TC, per ingress interface traffic counter (PSID.INT.I.TC)

This counter provides per Traffic Class (TC) breakdown of PSID.INT.I. It is defined as follows:

For a given Traffic Class (TC), ingress interface (INT.I), and a remote prefix SID S, N MAY maintain counter(s) for the traffic received on I and forwarded towards S.

13.4. Traffic Matrix Counters

A Traffic Matrix (TM) provides, for every ingress point N into the network and every egress point M out of the network, the volume of traffic T(N, M) from N to M over a given time interval. To measure the traffic matrix, nodes in an SR network designate its interfaces as either internal or external.

When Node N receives a packet destined to remote prefix SID M, N maintains the following counters. These counters include accounting due to push, pop or swap operations.

13.4.1. Per-Prefix SID Traffic Matrix counter (PSID.E.TM)

This counter is defined as follows:

For a given remote prefix SID M, N SHOULD maintain counter(s) for all the traffic received on any external interfaces and forwarded towards M.

13.4.2. Per-Prefix, Per TC SID Traffic Matrix counter (PSID.E.TM.TC)

This counter provides per Traffic Class (TC) breakdown of PSID.E.TM. It is defined as follows:

For a given Traffic Class (TC) and a remote prefix SID M, N SHOULD maintain counter(s) for all the traffic received on any external interfaces and forwarded towards M.

13.5. SR Policy Counters

Per policy counters are only maintained at the policy head-end node. For each SR policy, the head-end node maintains the following counters.

13.5.1. Per-SR Policy Aggregate traffic counter (POL)

This counter includes both labelled and unlabelled steered traffic. It is defined as:

For each SR policy (P), head-end node N MUST maintain counter(s) for the aggregate traffic steered onto P.

13.5.2. Per-SR Policy labelled steered aggregate traffic counter (POL.BSID)

This counter is defined as:

For each SR policy (P), head-end node N SHOULD maintain counter(s) for the aggregate labelled traffic steered onto P. Please note that labelled steered traffic refers to incoming packets with an active SID matching a local BSID of an SR policy at the head-end.

13.5.3. Per-SR Policy, per TC Aggregate traffic counter (POL.TC)

This counter provides per Traffic Class (TC) breakdown of POL. It is defined as follows:

For each SR policy (P) and a given Traffic Class (TC), head-end node N SHOULD maintain counter(s) for the aggregate traffic (matching the traffic class TC criteria) steered onto P.

13.5.4. Per-SR Policy, per TC labelled steered aggregate traffic counter (POL.BSID.TC)

This counter provides per Traffic Class (TC) breakdown of POL.BSID. It is defined as follows:

For each SR policy (P) and a given Traffic Class (TC), head-end node N MAY maintain counter(s) for the aggregate labelled traffic steered onto P.

13.5.5. Per-SR Policy, Per-Segment-List Aggregate traffic counter (POL.SL)

This counter is defined as:

For each SR policy (P) and a given Segment-List (SL), head-end node N SHOULD maintain counter(s) for the aggregate traffic steered onto the Segment-List (SL) of P.

13.5.6. Per-SR Policy, Per-Segment-List labelled steered aggregate traffic counter (POL.SL.BSID)

This counter is defined as:

For each SR policy (P) and a given Segment-List (SL), head-end node N MAY maintain counter(s) for the aggregate labelled traffic steered onto the Segment-List SL of P. Please note that labelled steered traffic refers to incoming packets with an active SID matching a local BSID of an SR policy at the head-end.

14. Appendix A

14.1. SRTE headend architecture

```
                                   +--------+  +--------+
                                   |  BGP   |  |  PCEP  |
                                   +--------+  +--------+
                                            \ /
                              +--------+  +--------+  +--------+
                              |  CLI   |--|  SRTE  |--| NETCONF|
                              +--------+  +--------+  +--------+
                                              |
                                          +--------+
                                          |  FIB   |
                                          +--------+
```

Figure 4: SRTE Architecture at a Headend

The SRTE functionality at a headend can be implemented in an SRTE process as illustrated in Figure 1.

The SRTE process interacts with other processes to learn candidate paths.

The SRTE process selects the active path of an SR Policy.

The SRTE process interacts with the RIB/FIB process to install an active SR Policy in the dataplane.

In order to validate explicit candidate paths and compute dynamic candidate paths, the SRTE process maintains an SRTE-DB. The SRTE process interacts with other processes (Figure 2) to collect the SRTE-DB information.

```
                                  +--------+  +--------+
                                  | BGP-LS |  |  IGP   |
                                  +--------+  +--------+
                                           \ /
                            +--------+  +--------+  +--------+
                            |   PCEP |--|  SRTE  |--| NETCONF|
                            +--------+  +--------+  +--------+
```

Figure 5: Topology/link-state database architecture

The SRTE architecture supports both centralized and distributed control-plane.

14.2. Distributed and/or Centralized Control Plane

14.2.1. Distributed Control Plane within a single Link-State IGP area

Consider a single-area IGP with per-link latency measurement and advertisement of the measured latency in the extended-TE IGP TLV.

A head-end H is configured with a single dynamic candidate path for SR policy P with a low-latency optimization objective and endpoint E.

Clearly the SRTE process at H learns the topology (and extended TE latency information) from the IGP and computes the solution SID list providing the low-latency path to E.

No centralized controller is involved in such a deployment.

The SRTE-DB at H only uses the Link-State DataBase (LSDB) provided by the IGP.

14.2.2. Distributed Control Plane across several Link-State IGP areas

Consider a domain D composed of two link-state IGP single-area instances (I1 and I2) where each sub-domain benefits from per-link latency measurement and advertisement of the measured latency in the related IGP. The link-state information of each IGP is advertised

via BGP-LS towards a set of BGP-LS route reflectors (RR). H is a headend in IGP I1 sub-domain and E is an endpoint in IGP I2 sub-domain.

Thanks to a BGP-LS session to any BGP-LS RR, H's SRTE process may learn the link-state information of the remote domain I2. H can thus compute the low-latency path from H to E as a solution SID list that spans the two domains I1 and I2.

The SRTE-DB at H collects the LSDB from both sub-domains (I1 and I2).

No centralized controller is required.

14.2.3. Centralized Control Plane

Considering the same domain D as in the previous section, let us know assume that H does not have a BGP-LS session to the BGP-LS RR's. Instead, let us assume a controller "C" has at least one BGP-LS session to the BGP-LS RR's.

The controller C learns the topology and extended latency information from both sub-domains via BGP-LS. It computes a low-latency path from H to E as a SID list <S1, S2, S3> and programs H with the related explicit candidate path.

The headend H does not compute the solution SID list (it cannot). The headend only validates the received explicit candidate path. Most probably, the controller encodes the SID's of the SID-List with Type-1. In that case, The headend's validation simply consists in resolving the first SID on an outgoing interface and next-hop.

The SRTE-DB at H only uses the LSDB provided by the IGP I1.

The SRTE-DB of the controller collects the LSDB from both sub-domains(I1 and I2).

14.2.4. Distributed and Centralized Control Plane

Consider the same domain D as in the previous section.

H's SRTE process is configured to associate color C1 with a low-latency optimization objective.

H's BGP process is configured to steer a Route R/r of extended-color community C1 and of next-hop N via an SR policy (N, C1).

Upon receiving a first BGP route of color C1 and of next-hop N, H recognizes the need for an SR Policy (N, C1) with a low-latency

objective to N. As N is outside the SRTE DB of H, H requests a controller to compute such SID list (e.g., PCEP).

This is an example of hybrid control-plane: the BGP distributed control plane signals the routes and their TE requirements. Upon receiving these BGP routes, a local headend either computes the solution SID list (entirely distributed when the endpoint is in the SRTE DB of the headend) else delegates the computation to a controller (hybrid distributed/centralized control-plane).

The SRTE-DB at H only uses the LSDB provided by the IGP.

The SRTE-DB of the controller collects the LSDB from both sub-domains.

14.3. Examples of Candidate Path Selection

Example 1:

Consider headend H where two candidate paths of the same SR Policy <color, endpoint> are signaled via BGP and whose respective NLRIs have the same route distinguishers:

NLRI A with distinguisher = RD1, color = C, endpoint = N, preference P1.

NLRI B with distinguisher = RD2, color = C, endpoint = N, preference P2.

o Because the NLRIs are identical (same distinguisher), BGP will perform bestpath selection. Note that there are no changes to BGP best path selection algorithm.
o H installs one advertisement as bestpath into the BGP table.
o A single advertisement is passed to the SRTE process.
o SRTE process does not perform any path selection.

Note that the candidate path's preference value does not have any effect on the BGP bestpath selection process.

Example 2:

Consider headend H where two candidate paths of the same SR Policy <color, endpoint> are signaled via BGP and whose respective NLRIs have different route distinguishers:

NLRI A with distinguisher = RD1, color = C, endpoint = N, preference P1.

NLRI B with distinguisher = RD2, color = C, endpoint = N, preference P2.

- o Because the NLRIs are different (different distinguisher), BGP will not perform bestpath selection.
- o H installs both advertisements into the BGP table.
- o Both advertisements are passed to the SRTE process.
- o SRTE process at H selects the candidate path advertised by NLRI B as the active path for the SR policy since P2 is greater than P1.

Note that the recommended approach is to use NLRIs with different distinguishers when several candidate paths for the same SR Policy (endpoint, color) are signaled via BGP to a headend.

Example 3:

Consider that a headend H learns two candidate paths of the same SR Policy <color, endpoint> one signaled via BGP and another via Local configuration.

NLRI A with distinguisher = RD1, color = C, endpoint = N, preference P1.

Local "foo" with color = C, endpoint = N, preference P2.

- o H installs NLRI A into the BGP table.
- o NLRI A and "foo" are both passed to the SRTE process.
- o SRTE process at H selects the candidate path indicated by "foo" as the active path for the SR policy since P2 is greater than P1.

When an SR Policy has multiple valid candidate paths with the same best preference, the SRTE process at a headend uses the rules described in section 2.9 to select the active path as explained in the following examples:

Example 4:

Consider headend H with two candidate paths of the same SR Policy <color, endpoint> and the same preference value both received from the same controller R and where RD2 is higher than RD1

- o NLRI A with distinguisher RD1, color C, endpoint N, preference P1(selected as active path at time t0).
- o NLRI B with distinguisher RD2 (RD2 is greater than RD1), color C, endpoint N, preference P1 (passed to SRTE process at time t1).

After t1, SRTE process at H selects candidate path associated with NLRI B as active path of the SR policy since RD2 is higher than RD1.

Note that, in such a scenario where there are redundant sessions to the same controller, the recommended approach is to use the same RD value for conveying the same candidate paths and let the BGP best path algorithm pick the best path.

Example 5:

Consider headend H with two candidate paths of the same SR Policy <color, endpoint> and the same preference value both received from the same controller R and where RD2 is higher than RD1.

Consider also that headend H is configured to override the discriminator tiebreaker specified in section 2.9

- o NLRI A with distinguisher RD1, color C, endpoint N, preference P1 (selected as active path at time t0).
- o NLRI B with distinguisher RD2, color C, endpoint N, preference P1 (passed to SRTE process at time t1).

Even after t1, SRTE process at H retains candidate path associated with NLRI A as active path of the SR policy since the discriminator tiebreaker is disabled at H.

Example 6:

Consider headend H with two candidate paths of the same SR Policy <color, endpoint> and the same preference value.

- o Local "foo" with color C, endpoint N, preference P1 (selected as active path at time t0).
- o NLRI A with distinguisher RD1, color C, endpoint N, preference P1 (passed to SRTE process at time t1).

Even after t1, SRTE process at H retains candidate path associated with local candidate path "foo" as active path of the SR policy since the Local protocol is preferred over BGP by default based on its higher protocol identifier value.

Example 7:

Consider headend H with two candidate paths of the same SR Policy <color, endpoint> and the same preference value but received via NETCONF from two controllers R and S (where S > R)

- o Path A from R with distinguisher D1, color C, endpoint N, preference P1 (selected as active path at time t0).

- Path B from S with distinguisher D2, color C, endpoint N, preference P1 (passed to SRTE process at time t1).

Note that the NETCONF process sends both paths to the SRTE process since it does not have any tiebreaker logic. After t1, SRTE process at H selects candidate path associated with Path B as active path of the SR policy.

14.4. More on Dynamic Path

14.4.1. Optimization Objective

This document defines two optimization objectives:

- Min-Metric - requests computation of a solution SID-List optimized for a selected metric.
- Min-Metric with margin and maximum number of SIDs - Min-Metric with two changes: a margin of by which two paths with similar metrics would be considered equal, a constraint on the max number of SIDs in the SID-List.

The "Min-Metric" optimization objective requests to compute a solution SID-List such that packets flowing through the solution SID-List use ECMP-aware paths optimized for the selected metric. The "Min-Metric" objective can be instantiated for the IGP metric xor the TE metric xor the latency extended TE metric. This metric is called the O metric (the optimized metric) to distinguish it from the IGP metric. The solution SID-List must be computed to minimize the number of SIDs and the number of SID-Lists.

If the selected O metric is the IGP metric and the headend and tailend are in the same IGP domain, then the solution SID-List is made of the single prefix-SID of the tailend.

When the selected O metric is not the IGP metric, then the solution SID-List is made of prefix SIDs of intermediate nodes, Adjacency SIDs along intermediate links and potentially BSIDs of intermediate policies.

In many deployments there are insignificant metric differences between mostly equal path (e.g. a difference of 100 usec of latency between two paths from NYC to SFO would not matter in most cases). The "Min-Metric with margin" objective supports such requirement.

The "Min-Metric with margin and maximum number of SIDs" optimization objective requests to compute a solution SID-List such that packets flowing through the solution SID-List do not use a path whose

cumulative O metric is larger than the shortest-path O metric + margin.

If this is not possible because of the number of SIDs constraint, then the solution SID-List minimizes the O metric while meeting the maximum number of SID constraints.

14.4.2. Constraints

The following constraints can be defined:

- o Inclusion and/or exclusion of TE affinity.
- o Inclusion and/or exclusion of IP address.
- o Inclusion and/or exclusion of SRLG.
- o Inclusion and/or exclusion of admin-tag.
- o Maximum accumulated metric (IGP, TE and latency).
- o Maximum number of SIDs in the solution SID-List.
- o Maximum number of weighted SID-Lists in the solution set.
- o Diversity to another service instance (e.g., link, node, or SRLG disjoint paths originating from different head-ends).

14.4.3. SR Native Algorithm

```
                       1----------------2----------------3
                      |\                               /
                      | \                             /
                      |  4-------------5-------------7
                      |   \                         /|
                      |    +-----------6-----------+ |
                      8------------------------------9
```

Figure 6: Illustration used to describe SR native algorithm

Let us assume that all the links have the same IGP metric of 10 and let us consider the dynamic path defined as: Min-Metric(from 1, to 3, IGP metric, margin 0) with constraint "avoid link 2-to-3".

A classical circuit implementation would do: prune the graph, compute the shortest-path, pick a single non-ECMP branch of the ECMP-aware shortest-path and encode it as a SID-List. The solution SID-List would be <4, 5, 7, 3>.

An SR-native algorithm would find a SID-List that minimizes the number of SIDs and maximize the use of all the ECMP branches along the ECMP shortest path. In this illustration, the solution SID-List would be <7, 3>.

In the vast majority of SR use-cases, SR-native algorithms should be preferred: they preserve the native ECMP of IP and they minimize the dataplane header overhead.

In some specific use-case (e.g. TDM migration over IP where the circuit notion prevails), one may prefer a classic circuit computation followed by an encoding into SIDs (potentially only using non-protected Adj SIDs to reflect the TDM paradigm).

SR-native algorithms are a local node behavior and are thus outside the scope of this document.

14.4.4. Path to SID

Let us assume the below diagram where all the links have an IGP metric of 10 and a TE metric of 10 except the link AB which has an IGP metric of 20 and the link AD which has a TE metric of 100. Let us consider the min-metric(from A, to D, TE metric, margin 0).

```
                                                  B---C
                                                  |   |
                                                  A---D
```

Figure 7: Illustration used to describe path to SID conversion

The solution path to this problem is ABCD.

This path can be expressed in SIDs as <B, D> where B and D are the IGP prefix SIDs respectively associated with nodes B and D in the diagram.

Indeed, from A, the IGP path to B is AB (IGP metric 20 better than ADCB of IGP metric 30). From B, the IGP path to D is BCD (IGP metric 20 better than BAD of IGP metric 30).

While the details of the algorithm remain a local node behavior, a high-level description follows: start at the headend and find an IGP prefix SID that leads as far down the desired path as possible(without using any link not included in the desired path). If no prefix SID exists, use the Adj SID to the first neighbor along the path. Restart from the node that was reached.

14.5. Benefits of Binding SID

The Binding SID (BSID) is fundamental to Segment Routing. It provides scaling, network opacity and service independence.

```
                       A---DCI1----C----D----E----DCI3---H
                      /            |         |            \
                     S             |         |             Z
                      \            |         |            /
                       B---DCI2----F---------G----DCI4---K
                    <==DC1==><=========Core========><==DC2==>
```

Figure 8: A Simple Datacenter Topology

A simplified illustration is provided on the basis of the previous diagram where it is assumed that S, A, B, Data Center Interconnect DCI1 and DCI2 share the same IGP-SR instance in the data-center 1 (DC1). DCI1, DCI2, C, D, E, F, G, DCI3 and DCI4 share the same IGP-SR domain in the core. DCI3, DCI4, H, K and Z share the same IGP-SR domain in the data-center 2 (DC2).

In this example, it is assumed no redistribution between the IGP's and no presence of BGP. The inter-domain communication is only provided by SR through SR Policies.

The latency from S to DCI1 equals to DCI2. The latency from Z to DCI3 equals to DCI4. All the intra-DC links have the same IGP metric 10.

The path DCI1, C, D, E, DCI3 has a lower latency and lower capacity than the path DCI2, F, G, DCI4.

The IGP metrics of all the core links are set to 10 except the links D-E which is set to 100.

A low-latency multi-domain policy from S to Z may be expressed as <DCI1, BSID, Z> where:

- o DCI1 is the prefix SID of DCI1.
- o BSID is the Binding SID bound to an SR policy <D, D2E, DCI3> instantiated at DCI1.
- o Z is the prefix SID of Z.

Without the use of an intermediate core SR Policy (efficiently summarized by a single BSID), S would need to steer its low-latency flow into the policy <DCI1, D, D2E, DCI3, Z>.

The use of a BSID (and the intermediate bound SR Policy) decreases the number of segments imposed by the source.

A BSID acts as a stable anchor point which isolates one domain from the churn of another domain. Upon topology changes within the core of the network, the low-latency path from DCI1 to DCI3 may change. While the path of an intermediate policy changes, its BSID does not change. Hence the policy used by the source does not change, hence the source is shielded from the churn in another domain.

A BSID provides opacity and independence between domains. The administrative authority of the core domain may not want to share information about its topology. The use of a BSID allows keeping the service opaque. S is not aware of the details of how the low-latency service is provided by the core domain. S is not aware of the need of the core authority to temporarily change the intermediate path.

14.6. Centralized Discovery of available SID in SRLB

This section explains how controllers can discover the local SIDs available at a node N so as to pick an explicit BSID for a SR Policy to be instantiated at headend N.

Any controller can discover the following properties of a node N (e.g., via BGP-LS, NETCONF etc.):

- o its local Segment Routing Label Block (SRLB).
- o its local topology.
- o its topology-related SIDs (Adj SID and EPE SID).
- o its SR Policies and their BSID ([I-D.ietf-idr-te-lsp-distribution]).

Any controller can thus infer the available SIDs in the SRLB of any node.

As an example, a controller discovers the following characteristics of N: SRLB [4000, 8000], 3 Adj SIDs (4001, 4002, 4003), 2 EPE SIDs (4004, 4005) and 3 SRTE policies (whose BSIDs are respectively 4006, 4007 and 4008). This controller can deduce that the SRLB sub-range [4009, 5000] is free for allocation.

A controller is not restricted to use the next numerically available SID in the available SRLB sub-range. It can pick any label in the subset of available labels. This random pick make the chance for a collision unlikely.

An operator could also sub-allocate the SRLB between different controllers (e.g. [4000-4499] to controller 1 and [4500-5000] to controller 2).

Inter-controller state-synchronization may be used to avoid/detect collision in BSID.

All these techniques make the likelihood of a collision between different controllers very unlikely.

In the unlikely case of a collision, the controllers will detect it through system alerts, BGP-LS reporting ([I-D.ietf-idr-te-lsp-distribution]) or PCEP notification. They then have the choice to continue the operation of their SR Policy with the dynamically allocated BSID or re-try with another explicit pick.

Note: in deployments where PCE Protocol (PCEP) is used between head-end and controller (PCE), a head-end can report BSID as well as policy attributes (e.g., type of disjointness) and operational and administrative states to controller. Similarly, a controller can also assign/update the BSID of a policy via PCEP when instantiating or updating SR Policy.

15. Acknowledgement

The authors like to thank Tarek Saad and Dhanendra Jain for their valuable comments and suggestions.

16. Normative References

[GLOBECOM]
Filsfils, C., Nainar, N., Pignataro, C., Cardona, J., and P. Francois, "The Segment Routing Architecture, IEEE Global Communications Conference (GLOBECOM)", 2015.

[I-D.ietf-idr-te-lsp-distribution]
Previdi, S., 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-08 (work in progress), December 2017.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and J. Tantsura, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-15 (work in progress), December 2017.

[I-D.ietf-pce-pce-initiated-lsp]
Crabbe, E., Minei, I., Sivabalan, S., and R. Varga, "PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model", draft-ietf-pce-pce-initiated-lsp-11 (work in progress), October 2017.

[I-D.ietf-pce-segment-routing]
Sivabalan, S., Filsfils, C., Tantsura, J., Henderickx, W., and J. Hardwick, "PCEP Extensions for Segment Routing", draft-ietf-pce-segment-routing-11 (work in progress), November 2017.

[I-D.ietf-pce-stateful-pce]
Crabbe, E., Minei, I., Medved, J., and R. Varga, "PCEP Extensions for Stateful PCE", draft-ietf-pce-stateful-pce-21 (work in progress), June 2017.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-15 (work in progress), January 2018.

[I-D.previdi-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-previdi-idr-segment-routing-te-policy-07 (work in progress), June 2017.

[I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Previdi, S., Tantsura, J., Hardwick, J., and D. Dhody, "Carrying Binding Label/Segment-ID in PCE-based Networks.", draft-sivabalan-pce-binding-label-sid-03 (work in progress), July 2017.

[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>.

[SIGCOMM] Hartert, R., Vissicchio, S., Schaus, P., Bonaventure, O., Filsfils, C., Telkamp, T., and P. Francois, "A Declarative and Expressive Approach to Control Forwarding Paths in Carrier-Grade Networks, ACM SIGCOMM", 2015.

Authors' Addresses

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

Email: cfilsfil@cisco.com

Siva Sivabalan
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, Ontario K2K 3E8
Canada

Email: msiva@cisco.com

Kamran Raza
Cisco Systems, Inc.
2000 Innovation Drive
Kanata, Ontario K2K 3E8
Canada

Email: skraza@cisco.com

Jose Liste
Cisco Systems, Inc.
821 Alder Drive
Milpitas, California 95035
USA

Email: jliste@cisco.com

Francois Clad
Cisco Systems, Inc.

Email: fclad@cisco.com

Ketan Talaulikar
Cisco Systems, Inc.

Email: ketant@cisco.com

Zafar Ali
Cisco Systems, Inc.

Email: zali@cisco.com

Shraddha Hegde
Juniper Networks, Inc.
Embassy Business Park
Bangalore, KA 560093
India

Email: shraddha@juniper.net

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

Email: daniel.voyer@bell.ca

Steven Lin
Google, Inc.

Email: stevenlin@google.com

Alex Bogdanov
Google, Inc.

Email: bogdanov@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

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

Email: pamattes@microsoft.com