Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 22, 2017

C. Filsfils
S. Sivabalan
Cisco Systems, Inc.
D. Yoyer
Bell Canada.
M. Nanduri
Microsoft Corporation.
S. Lin
A. Bogdanov
Google, Inc.
M. Horneffer
Deutsche Telekom
F. Clad
Cisco Systems, Inc.,
D. Steinberg
Steinberg Consulting
B. Decraene
S. Litkosky
Orange Business Services

February 18, 2017

Segment Routing Policy for Traffic Engineering
draft-filsfils-spring-segment-routing-policy-00.txt

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 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

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 Traffic Engineering Architecture

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

Figure 1: SR Policy architecture

The Segment Routing Traffic Engineering (SRTE) process installs a Segment Routing Policy (SR Policy) in the forwarding plane (FIB).

An SR policy is represented in FIB as a BSID-keyed entry with the action of steering the packets matching this entry to the selected path of the SR Policy.

For a given SR policy, the SRTE process MAY learn multiple candidate paths from different sources: NETCONF with OpenConfig or YANG model (work in progress), PCEP [I-D.ietf-pce-pce-initiated-lsp], local configuration or BGP [I-D.previdi-idr-segment-routing-te-policy].

The SRTE process selects the best candidate path and installs it in FIB.

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

Figure 2: Topology/link-state database architecture

The SRTE process maintains an SRTE database (SRTE-DB).

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.

3. SR Policy

An SR Policy is identified through the following tuple:

o The head-end where the policy is instantiated/implemented.
o The endpoint (i.e.: the destination of the policy).
o The color (an arbitrary numerical value).

At a given head-end, an SR Policy is fully identified by the <color, endpoint> tuple.

An endpoint can be specified as an IPv4 or IPv6 address.

An SR Policy contain contains one or more candidate paths.

An SR Policy instantiates one single path in RIB/FIB: i.e. the selected path among the candidate paths.

A candidate path is either dynamic or explicit.

A dynamic path expresses an optimization objective and a set of constraints. The headend computes a solution to the optimization problem as a Segment Identifier (SID) list or a set of SID lists. When the headend does not have enough topological information (e.g. multi-domain problem), the headend may delegate the computation to a PCE. Whenever the network situation changes, the path is recomputed.

An explicit path is a SID list or a set of SID lists.

A candidate path has a preference. If not specified, the default preference is 100.

A candidate path is associated with a single Binding SID (BSID).

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 defined in a later section.

A Path is selected (i.e. it is the best path of the policy) when it is valid and its preference is the best (highest value) among all the paths of the SR Policy.

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

A headend may be informed about a path for a policy <color, endpoint> by various means including: local configuration, NETCONF, PCEP or BGP. The protocol source of the path does not matter to the path selection logic.

In the vast majority of use-cases known to date, a path is associated with a single SID list and each path of a policy has a different preference.

The SID list of an SR Policy is the SID list of its selected path.

The BSID of an SR Policy refers to its selected path.

In all the use-cases known to date, all the paths associated with a given policy have the same BSID. One may thus assume that in practice a policy has a stable BSID that is independent of the selected path changes and this BSID is an identification of a policy. However, one should know that a BSID MAY change over the life of an SR Policy and the true identification of a policy is the tuple <headend, endpoint, color>.

An SR Policy <color, endpoint> is active at a headend as soon as this head-end knows about a valid path for this policy.

An active SR Policy installs a BSID-keyed entry in the forwarding plane with the action of steering the packets matching this entry to the SID list of the SR Policy.

If a set of SID lists is associated with the selected path of the policy, then the steering is flow and W-ECMP based according to the relative weight of each SID list.

In summary, the information model is the following:

```
         SR policy FOO
             path 200 (selected)
                 BSID1
                 Weight W1, SID list1: SID11...SID1i
                 Weight W2, SID list2: SID21...SID2j
             path 100 (selected)
                 BSID2
                 Weight W3, SID list3: SID31...SID3i
                 Weight W4, SID list4: SID41...SID4j
```

In general BSDIn = BSID1 = BSID2 ...

4. SID List

The segment list (SID list) includes segments of different types (1 to 8) and an optional weight value that is used for W-ECMP.

The following segment types are defined:

Type 1:	SID only, in the form of MPLS Label.
Type 2:	SID only, in the form of IPv6 address.
Type 3:	IPv4 Node Address with optional SID.
Type 4:	IPv6 Node Address with optional SID.
Type 5:	IPv4 Address + index with optional SID.
Type 6:	IPv4 Local and Remote addresses with optional SID.
Type 7:	IPv6 Address + index with optional SID.
Type 8:	IPv6 Local and Remote addresses with optional SID.

The optional SID can be an MPLS label (SR applied to the MPLS dataplane) or an IPv6 SID (SRv6, SR applied to the IPv6 dataplane).

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.

A SID list is represented as <S1, S2, ... Sn> where S1 is the first SID.

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 SRTE 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.

5. SR Policy Multi-Domain Database

A headend can learn an attached domain topology via its IGP or a BGP-LS session. A headend can learn a non-attached domain topology via a BGP-LS session.

A headend collects all these topologies in the SR-TE database (SRTE-DB).

The SRTE-DB is multi-domain capable.

In some deployments, the SRTE-DB may only contain the attached domain topology while in others, the SRTE-DB may contain the topology of multiple domains.

6. Operations

6.1. W-ECMP

Packets steered to an SR Policy (i.e. to its BSID either via presence in the packet header as active segment or via FIB recursion) are load-balanced on a weighted basis among the SID lists associated with the selected path of the SR Policy.

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.

6.2. Path Validation

A SID List is invalid as soon as:

- o It is empty.
- 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-8 into an MPLS label or an SRv6 SID.

Unreachable 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 whatever is type.

A Path is invalid as soon as it has no valid SID list.

The headend of an SR Policy updates the validity of a SID list upon network topological change.

A path of an SR Policy is invalid when all its SID lists are invalid.

An SR Policy is invalid when all its paths are invalid.

6.3. Fast Convergence

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 in which order the policies should be re-computed. Such a priority may be represented by an integer in the range [0, 254] where the lowest value is the highest priority.

7. Binding SID

7.1. Benefits

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 3: A Simple Datacenter Topology

A simplified illustration is provided on the basis of the previous diagram where we assume 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, we assume 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 SRTE 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.

7.2. Allocation

There are three approaches to allocate a BSID to an SR Policy: all the paths have no explicit BSID (called dynamic allocation), all the paths have the same explicit BSID (explicit allocation) and finally a mix of paths with and without explicit BSID (generic allocation).

In practice, all the use-cases seen to-date either use the explicit allocation or the dynamic allocation. The explicit allocation is most-often associated with controller-instantiated SR Policies. The dynamic allocation is most-often associated with router-based on-demand SR Policies.

7.2.1. Dynamic BSID Allocation

No path of the SR Policy have a specified BSID.

In such a case, the SR-TE implementation allocates a SID to the SR Policy and keeps it along the whole existence of the policy.

In the case of SR-MPLS, the SR-TE implementation binds a local dynamic label in the same way LDP, RSVP-TE or BGP would do.

7.2.2. Explicit BSID Allocation

All the paths of the SR Policy have the same specified BSID, with the same behavioral preference in case this specified BSID is not available.

If the specified BSID is available, then it is bound to the SR Policy and used along the existence of the policy.

If the specified BSID is not available, then a SYSLOG/NETCONF message is generated and if the preferred behavior is to fall-back on the dynamic allocation, then the dynamic allocation is performed.

If the specified BSID is not available and the operator-requested behavior is to not fall-back on the dynamic allocation, then a SYSLOG/NETCONF message is generated and the SR Policy does not install any BSID entry in the forwarding plane.

A later section will explain 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.

7.2.3. Generic BSID Allocation

This section details the BSID allocation when a policy is made of paths with different BSID allocation behaviors (e.g. mix of paths with and without an explicit BSID, potentially with different explicit BSIDs).

When the selected 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).

If the selected path's BSID is not available, then the SR Policy keeps the previous BSID. If the SR Policy did not have a previous BSID, then the SR Policy dynamically binds a BSID to itself.

Note that a path may request that only its specified BSID be used. In that case, if that BSID is not available and that path is active, then no BSID is bound to the policy and a SYSLOG/NETCONF is triggered. In this case, the SR Policy does not install any entry indexed by a BSID in the forwarding plane.

When an SR Policy has multiple multiple valid paths with the best preference but with different BSIDs, it is left to the implementation to decide which BSID to install. This case is unlikely in practice for two reasons. First, all known use-cases share the same BSID across all the paths of a given SR Policy. Second, all known use-cases have a different preference for each path. Hence in practice a single path will be active and with a stable BSID on a per-policy basis.

8. Centralized Discovery

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.):

- its local Segment Routing Label Block (SRLB).
- its local topology.
- its topology-related SIDs (Adj SID and EPE SID).
- 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.

Likely, the next question is: how do we ensure that different controllers do not pick the same available SID at the same time for different SR Policies.

Clearly, 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 SYSLOG/NETCONF, 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.

9. Dynamic Path

A dynamic path is a path that expresses an optimization objective and constraints.

The headend of the policy is responsible to compute a SID list ("solution SID list") that fits this optimization problem. The headend is responsible for computing the solution SID list any time the inputs to the problem change (e.g. topology changes).

9.1. Optimization Objective

We define two optimization objectives:

- o Min-Metric - requests computation of a solution SID list optimized for a selected metric.
- o 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 cumulated 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.

9.2. 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).

9.3. SR Native Algorithm

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

Figure 4: 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.

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

9.4. 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 5: Illustration used to describe path to SID conversion

The solution path to this problem is ABCD.

This path can be expressed in SIDs as $#60;B, D$#62; 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.

9.5. PCE Computed Path

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

10. Signaling Paths of an SR Policy to a Head-end

A headend H can be informed about a path for an SR policy (endpoint, color) via several means: BGP, PCEP, CLI, netconf.

We remind that the selection of the best path for a policy is independent of the protocol source of the path.

10.1. BGP

Please refer to [I-D.previdi-idr-segment-routing-te-policy]

10.2. PCEP

Please refer to [I-D.ietf-pce-pce-initiated-lsp]

10.3. NETCONF

Operator MUST be able to install policy via NETCONF with OpenConfig/ YANG models (work in progress).

10.4. CLI

Operator MUST be able to install policy via CLI.

11. Steering into an SR Policy

A headend can steer a packet flow on an SR Policy in various ways:

- o Incoming packets have an active SID matching a local BSID at the head-end.
- o Incoming packets match a BGP/Service route which recurses on the BSID of a local policy.
- o Incoming packets match a BGP/Service route which recurses on an array of paths to the BGP nhop where some of the paths in the array are local SR Policies.
- o Incoming packets match a routing policy which directs them on a local SR policy.

For simplicity of illustration, we will use the SR-MPLS example.

11.1. Incoming Active SID is a BSID

Let us assume that headend H has a local SR Policy P of SID list <S1, S2, S3> and BSID B.

When H receives a packet with label stack <B, L2, L3>, H pops B and pushes <S1, S2, S3>. H sends the resulting packet with label stack <S1, S2, S3, L2, L3> along the path to S1.

H has steered the packet in the 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 such as to better scale and simplify intermediate aggregation nodes.

11.2. Recursion on a BSID

Let us assume that headend H:

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

In such a case, H installs R/r in RIB/FIB with next-hop = B (instead of N).

Indeed, H's local BGP policy and the received BGP route indicate that the headend should associate R/r with an SR-TE 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 nhop and installing the BGP route on that generalized next-hop.

When H receives a packet 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 label associated with the BGP route is pushed after the SID list of the SR Policy.

11.3. Recursion on a dynamic BSID

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

In this section, we note that this policy may be generated dynamically by the head-end H upon reception of the BGP route R/r via N with color C.

A possible implementation has the BGP policy matches on the color C and triggers an on-demand local request to the SR-TE process to instantiate an SR Policy (endpoint N, color C). Color C is bound to some optimization objective and constraints specified in the local BGP policy defined for color C. Once the related SR Policy is instantiated, the SR-TE process returns the related BSID to BGP process which can then installs the BGP route R/r on B.

The rest of the explanation is the same as the previous section.

11.4. An array of BSIDs associated with an IGP entry

Let us assume that head-end H:

- o learns about a BGP route R/r via next-hop N and label V.
- o has a local SR Policy P1 to (endpoint = N, color = C1) of SID list <S1, S2, S3> and BSID B1.
- o has a local 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.
- o 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).

In such a case, H installs in RIB/FIB:

- o R/r in with next-hop N (as usual).
- o N via a recursion on an array A (instead of the immediate outgoing link associated with the IGP shortest-path to N.
- o Entry A(0) set to the immediate outgoing link of the IGP shortest-path to N.
- o Entry A(1) set to B1.
- o Entry A(2) set to B2.

H receives three packets P, P1 and P2 on its incoming interface. H colors them respectively with forwarding-class 0, 1 and 2. As a result:

o H pushes <V> on packet P and forwards the resulting frame along the shortest-path to N (which in SR-MPLS results in the pushing of the prefix-SID of N.
o H pushes <S1, S2, S3, V> on packet P1 and forwards the resulting frame along the shortest-path to S1.
o H pushes <S4, S5, S6, V> on packet P2 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).

This realizes per-flow steering: different flows bound to the same BGP destination R/r are steered on different SR-TE paths.

11.5. A Routing Policy on a BSID

Finally, headend H may be configured with a local routing policy which overrides any BGP/IGP path and steer a specified flow on an SR Policy.

12. Optional Steering Modes for BGP Destinations

12.1. Color-Only BGP Destination Steering

In the previous section "Recursion on a BSID", we have 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 a local SR Policy defined by the tuple (N, C).

This is the most likely form of BGP destination steering and the one we recommend.

In this section, we define 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.

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

When 00, the BGP destination is preferably steered onto a valid SR Policy (N, C) where N is an IPv4/6 endpoint address and C is a color

value else it is steered on the IGP path to the next-hop N. This is the classic case we described before and that we recommend.

When 01, the BGP destination is preferably steered onto a valid SR Policy (N, C) else onto a valid SR Policy (null endpoint, C) else on the IGP path to the next-hop N.

When 10, the BGP destination is preferably steered onto a valid SR Policy (N, C) else onto a valid SR Policy (null endpoint, C) else on any valid SR Policy (any endpoint, C) else 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).

When 11, it is treated like 00.

12.2. Drop on Invalid

The local BGP policy authorizing the use of an extended color community steering on an SR policy may specify that if the related SR Policy becomes invalid then the related BSID should remain in RIB/FIB and point to null0 (drop any packet recursing on that BSID).

Recall that, by default, for a BGP route R/r via next-hop N with extended-color community C, when the SR Policy (N, C) becomes invalid, then BGP re-installs R/r in RIB/FIB via N (the IGP path to N).

13. Multipoint SR Policy

13.1. Spray SR Policy

A Spray SR-TE policy is a variant of an SR-TE 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.

14. Reporting SR Policy

Stateful PCEP ([I-D.ietf-pce-stateful-pce] and [I-D.sivabalan-pce-binding-label-sid] provides an ability for head-end to report BSID, attributes, and operational/administrative states. Using this protocol, a PCE can also update an existing SR Policy whose path computation is delegated to it as well as instantiate new SR Policy on a head-end.

BGP-LS reports an SR Policy via ([I-D.ietf-idr-te-lsp-distribution]

15. Work in Progress

- o Open configuration model.
- o Yang model.

16. Acknowledgement

17. 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. jefftant@gmail.com, "Distribution of Traffic Engineering (TE) Policies and State using BGP-LS", draft-ietf-idr-te-lsp-distribution-06 (work in progress), January 2017.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Filsfils, C., Bashandy, A., Gredler, H., Litkowski, S., Decraene, B., and j. jefftant@gmail.com, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-09 (work in progress), October 2016.

[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-07 (work in progress), July 2016.

[I-D.ietf-pce-segment-routing]
Sivabalan, S., Medved, J., Filsfils, C., Crabbe, E., Raszuk, R., Lopez, V., Tantsura, J., Henderickx, W., and J. Hardwick, "PCEP Extensions for Segment Routing", draft-ietf-pce-segment-routing-08 (work in progress), October 2016.

[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-18 (work in progress), December 2016.

[I-D.ietf-spring-segment-routing]
Filsfils, C., Previdi, S., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", draft-ietf-spring-segment-routing-11 (work in progress), February 2017.

[I-D.previdi-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Sreekantiah, A., Sivabalan, S., Mattes, P., Rosen, E., and S. Lin, "Advertising Segment Routing Traffic Engineering Policies in BGP", draft-previdi-idr-segment-routing-te-policy-03 (work in progress), December 2016.

[I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Previdi, S., Tantsura, J., Hardwick, J., and M. Nanduri, "Carrying Binding Label/Segment-ID in PCE-based Networks.", draft-sivabalan-pce-binding-label-sid-02 (work in progress), October 2016.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://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

Daniel Yoyer
Bell Canada.

Email: daniel.yoyer@bell.ca

Mohan Nanduri
Microsoft Corporation.
One Microsoft Way
Redmond, WA 98052
USA

Email: mnanduri@microsoft.com

Steven Lin
Google, Inc.

Email: stevenlin@google.com

Alex Bogdanov
Google, Inc.

Email: bogdanov@google.com

Martin Horneffer
Deutsche Telekom

Email: martin.horneffer@telekom.de

Francois Clad
Cisco Systems, Inc.,

Email: fclad@cisco.com

Dirk Steinberg
Steinberg Consulting

Email: dws@steinbergnet.net

Bruno Decraene
Orange Business Services

Email: bruno.decraene@orange.com

Stephane Litkosky
Orange Business Services

Email: stephane.litkowski@orange.com