Source Packet Routing in Networking
Internet-Draft
Intended status: Standards Track
Expires: 23 April 2026

A. Stone
Nokia
V. P. Beeram
Juniper Networks
N. Buraglio
Energy Sciences Network
S. Peng
ZTE Corporation

20 October 2025

Multipath Traffic Engineering for Segment Routing
draft-stone-spring-mpte-sr-01

Abstract

This document describes a mechanism to achieve Multipath Traffic Engineering for Segment Routing based networks.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Source Packet Routing in Networking Working Group mailing list (spring@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/spring/.

Source for this draft and an issue tracker can be found at https://github.com/astone282/draft-stone-spring-mpte-sr.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 23 April 2026.

Table of Contents

1. Introduction

The document [I-D.draft-kompella-teas-mpte] introduces a multipath traffic engineering concept that combines the benefits of both Equal-Cost Multipath (ECMP) forwarding and traffic-engineered paths. This approach uses a Directed Acyclic Graph (DAG) based forwarding mechanism, with the DAG signaled to participating network nodes. The concept is to move beyond simple ECMP paths by incorporating both ECMP and non-ECMP paths while still adhering to traffic engineering constraints, to provide added resiliency while also permitting better usage of link bandwidth.

[I-D.draft-kompella-teas-mpte] outlines the architecture design which can be applied to both distributed and centralized signaling for various tunnel types, including MPLS, IP, and others while leaving the specific details of each out of scope.

This document proposes and discusses a centralized computation and signaling mechanism for SR-based networks, primarily utilizing existing constructs and capabilities. As MPTE evolves, new extensions to SR-based documents may be needed, both in terms of architecture and protocol-specific semantics.

The document assumes the reader is familiar with [RFC8402], [RFC9256], and [I-D.draft-kompella-teas-mpte].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

* For MPTE terminology such as MPTED, DAG, MC, MID, Junction node and others see [I-D.draft-kompella-teas-mpte].

* For SR terminology see [RFC8402] and [RFC9256].

3. MP-TE vs Multiple SID lists

It's important to recognize the SR Policy information model supports multiple SID lists, effectively encoding multiple unique paths on a tunnel at the ingress. A Directed Acyclic Graph (DAG) can be represented as a collection of individual paths, each of which can be programmed as a separate SID list within an SR Policy Candidate Path. However, depending on the graph topology, the number of unique paths to encode can grow significantly. Additionally, in traditional SID list approaches, hashing is performed only at the ingress, rather than at each downstream node. In contrast, a DAG-based mechanism may allow better traffic distribution or localized tuning based on localized weight changes. Finally, the maximum segment depth (MSD) may need to be considered for long paths that deviate significantly from the shortest path.

In comparison, encoding a DAG' s forwarding instructions across the participating Junction nodes reduces the number of individual SID lists at the ingress, but at the cost of increasing state in the network. While source-based routing aims to reduce network state, there is a trade-off between the volume and length of SID lists versus distributing that state throughout the network to achieve multipath traffic engineering use cases.

The choice between using multiple ingress segment lists or the MPTE DAG-based distribution mechanism depends on the traffic engineering requirements, overall network design, link/path metrics, and the DAG' s structure.

4. MP-TE concepts with Segment Routing

This document proposes the below concepts for applying MPTE in an SR environment.

4.1. MPTED

The MPTED is managed by a centralized controller, such as a PCE acting as the MC. Topology discovery is performed using BGP-LS [RFC7752], while transport control plane signaling is achieved through controller-oriented protocols such as PCEP [RFC5440], [RFC8231] and BGP/BGP-LS [I-D.draft-ietf-idr-segment-routing-te-policy], [I-D.draft-ietf-idr-bgp-ls-sr-policy]. The MC computes, manages, and distributes all forwarding information to the nodes participating in the MPTE DAG, which form the MPTED.

[I-D.draft-kompella-teas-mpte] specifies that a node in the MPTED is identified by its IPv6 loopback address. However, this document allows the use of a 32-bit dotted quad router ID as an alternative. This value represents the headend address of a node participating in the DAG.

As per [I-D.draft-kompella-teas-mpte], the controller acting as the MC is responsible for assigning the MPID and incrementing the MPTED unique ID version.

4.2. Junction Segment

The concept of a Junction Segment is introduced to describe the signaling and forwarding behavior of a Junction node in an SR network.

It’s worth noting that the architectural use of a Junction Segment is analogous to a Replication Segment [RFC9524], but it performs forwarding based on load balancing rather than replication.

A Junction Segment is installed on nodes identified as Junction Nodes, as defined in [I-D.draft-kompella-teas-mpte].

This document version proposes that a Junction Segment is realized using the existing SR Policy construct with a single Candidate Path and a Binding SID.

A Binding Segment is attached to an SR Policy Candidate Path with one or more SID Lists. OIF instruction signaling is achieved via segment lists, where the top SID identifies the outgoing interface(s).

Since a Junction Segment may egress to multiple downstream nodes, the endpoint of the corresponding SR Policy MUST be set to the null value (0.0.0.0). Therefore, a Junction segment is identified by its <headend, color> attribute.

This document assumes that the color value is mapped to the <MID, Version> tuple and is tracked by the controller.

It is RECOMMENDED that a globally consistent color value be used across all Junction Segments that belong to a single DAG instance or that serve a common DAG intent.

A DAG may consist of multiple Junction Segments that collectively represent a single DAG tunnel. This MP-TE tunnel is used by one or many SR Policies instantiated on one or many ingress nodes. A Junction Segment may be reused by multiple ingress SR Policies, provided that the subgraph it forwards over is fully shared by all SR Policies using it, including the set of their respective egress nodes.

The ingress SR Policy is responsible for initiating traffic steering into the DAG and is associated with a color value representing service intent. The junction segment, on the other hand, is a DAG forwarding construct implemented as an SR Policy with a BSID.

To support global optimization with make-before-break (MBB) operations across a set of ingress SR Policies, the color value used by Junction Segments SHOULD differ from the color values of the ingress SR Policies that are consumers of the DAG. This distinction allows ingress SR Policies to independently manage service steering while enabling consistent forwarding behavior across the DAG.

The controller is responsible for maintaining and tracking the association and semantic meaning of color values across all Junction Segments that participate in the DAG. See Section 9.5 for additional discussion.

Accordingly, an implementation SHOULD treat ingress SR Policies and Junction Segments as decoupled constructs, each with their own versioning and lifecycle.

Since SR-based networks support specifying multiple egress interfaces using adjacency-SID sets and Node SIDs, the Junction Segment MAY include a SID list entry that identifies multiple outgoing interfaces. In addition to egress interfaces, [I-D.draft-kompella-teas-mpte] describes signaling ingress interfaces. The use of a Junction Segment omits the need for per-interface ingress signaling as a single Binding Segment attached to an SR Policy is used. All upstream originated traffic sent to a downstream Junction Node uses the same, single Junction Segment value which is a Binding Segment.

4.3. MPTE SR Policy - Tunnel with multiple ingress/egress

[I-D.draft-kompella-teas-mpte] specifies that an MPTE Tunnel could have multiple ingress and/or multiple egress nodes. Currently, the SR Policy architecture defines an SR Policy using a {Headend, Endpoint, Color} tuple, where the Endpoint may be set to the null value (0.0.0.0), indicating multiple destinations.

For controller-initiated tunnels, the intended ingress and egress node(s) can be provided to the controller based on implementation-specific methods. These may be signaled to the network as multiple tunnels to support multi-ingress scenarios. Each tunnel MAY use a null Endpoint value to support multi-egress.

However, MPTE SR Policies that are originated or defined by network devices are typically limited to a single ingress and a single egress endpoint unless protocols such as PCEP or NETCONF are extended to encode additional intended destination node(s) for controller-based path computation.

As mentioned in section 4.2, the DAG tunnel may be re-used by multiple ingress SR Policies. This mechanism is used to support acheiving multiple ingress nodes originated from the network, by way of the controller binding and attaching the ingress SR Policies to a pre-existing DAG sharing the same intent and endpoints.

5. Operation

A path computation request or tunnel delegation notification is sent to the controller, specifying one or more ingress and egress nodes, along with constraints. This request may originate from an ingress router in the network or be provisioned directly via an API to the controller. This tunnel computation request or delegation pertains to an instance of an MPTE Tunnel achieved with the ingress SR Policy..

The controller computes the DAG for ingress and all egress endpoints to determine all Junction nodes in the DAG to be used for the tunnel.

The controller signals the Junction Segment(s) to all downstream nodes, starting with the penultimate egress hop node(s) and working upwards toward the ingress nodes.

Junction Segment deployments are in the form of a unicast SR Policy with a single Candidate Path using protocols such as PCEP, BGP, or NETCONF. The optional use of an MPTED Reflector is protocol specific. For example, PCEP sessions terminate on each and every Junction node in the topology and BGP may do the same or make use of a BGP Route Reflector. A BSID MUST be explicitly requested or signaled to the Junction node for assignment. If the controller opts for local node assigned value, it MUST wait to signal upstream Junction nodes about their Junction segments in their outgoing SID lists. The BSID value MAY be a constant value globally, if assigned by PCE/Controller, or may be a different value on each Junction Node whether it’ s assigned by PCE/Controller or the local Junction Node itself.

Each SR Policy contains one or more SID lists. These SID lists must include at least two segment identifiers: one SID for forwarding to the downstream Junction node and one for the Junction Segment value (BSID) of the downstream node. For directly connected neighbors, this may be the adjacency or node SID for the neighbor. For Junction nodes that are not directly connected, additional SIDs MAY be used to steer the packet along an ECMP or non-ECMP path to the downstream Junction node. It is worth noting that if the SID list comprises of only an Adjacency-SID and the Junction SID of the neighbor node, then the dataplane packet contains only one SID on egress which is the Junction SID of the neighboring node.

5.1. Example

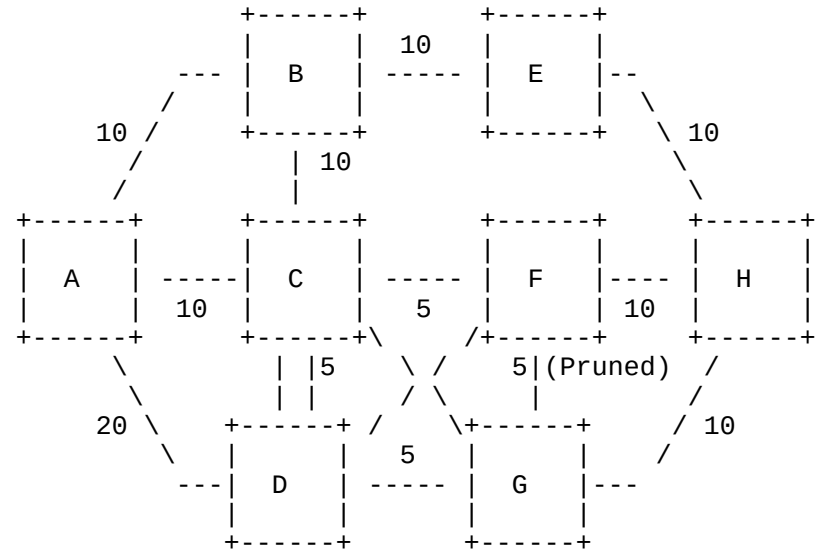

Figure 1 : Example Topology to apply DAG

Figure 1 presents a sample topology for one ingress and one egress MPTE Tunnel established as an SR Policy. The MPTE tunnel is from A to H. The figure presents the bi-directional link weights for an arbitrary metric (IGP, TE, Delay etc).

Note there is a mesh between C, D, F and G of weight 5 per link. Pruned represents an excluded link due to TE constraints. In the below, the terminology {u,v} represents a unidirectional link between U and V. The terminology Adj-SID-XY represents an adjacency SID from X to Y. Node-SID-X represents the node SID path to X.

A MPTE DAG is computed to contain nodes B,C,D,E,F,G with the links {G, F} / {F, G} pruned.

The below Junction Segments are deployed to the network realized with an SR Policy with a single Candidate Path containing multiple segment lists. Note the following:

* When the DAG is computed, loops cannot exist. Therefore, in the above topology the links in direction {C, B} and {C,D} and {C,F} and {C,G} are chosen in the DAG.

* The path along {B,E,H} can be represented by a single Node SID. A Junction on node E is not required. Junction Segment B may also be omitted (as per Section 8 below) but is utilized here for example purposes.

* Junction F is described below, but an optimization could be performed to exclude Junction F as it only has one egress link (see Section 8 below).

* In practice the Binding Segments MAY be all the same value. This example describes different BSID values for readability.

* Weights of each egress SID list is also currently omitted.

* The color on the ingress SR Policy still provides intent steering for ingress traffic, therefore SHOULD be unique to the color value of the Junction segments comprising of the DAG to permit globalized make-before-break behavior.

```
Junction Segment B:
  Color: 100
  BSID: BSID-B
  SID List 1: [Node-SID-H]

Junction Segment F:
  Color: 100
  BSID: BSID-F
  SID List 1: [Adj-SID-FH]

Junction Segment G:
  Color: 100
  BSID: BSID-G
  SID List 1: [Adj-SID-GH]

Junction Segment C:
  Color: 100
  BSID: BSID-C
  SID List 1: [Adj-SID-CB, BSID-B]
  SID List 2: [Adj-SID-CF, BSID-F]
  SID List 3: [Adj-SID-CG, BSID-G]
  SID List 4: [Node-SID-D, BSID-D]

Junction Segment D:
  Color: 100
  BSID: BSID-D
  SID List 2: [Adj-SID-DF, BSID-F]
  SID List 3: [Adj-SID-DG, BSID-G]
```

Then lastly, at ingress the SR Policy transport tunnel is configured with the following:

```
Ingress SR Policy (Using Junction Segments):
  Color: 50
  Candidate Path 1:
    SID List 1: [Adj-SID-AB, BSID-B]
    SID List 2: [Adj-SID-AC, BSID-C]
    SID List 3: [Adj-SID-AD, BSID-D]
```

In comparison, if the above DAG was encoded at ingress then the following individual segment lists could be used to represent the above DAG. Note, some of the below could be compressed with a Node SID(s) but listed with adjacency for explicit example.

```
Ingress SR Policy (Ingress only):
  Color: 50
  Candidate Path 1:
    SID List 1: [Adj-SID-AC, Adj-SID-CF, Adj-SID-FH]
    SID List 2: [Adj-SID-AC, Node-SID-D, Adj-SID-DG, Adj-SID-GH]
    SID List 3: [Adj-SID-AC, Node-SID-D, Adj-SID-DF, Adj-SID-FH]
    SID List 4: [Adj-SID-AC, Adj-SID-CF, Adj-SID-CG, Adj-SID-GH]
    SID List 5: [Adj-SID-AC, Adj-SID-CF, Adj-SID-BE, Adj-SID-EH]
    SID List 6: [Adj-SID-AB, Node-SID-H]
    SID List 7: [Adj-SID-AD, Adj-SID-DG, Adj-SID-GH]
    SID List 8: [Adj-SID-AD, Adj-SID-DF, Adj-SID-FH]
```

6. Load-balancing

When a packet with the BSID assigned to the Junction Segment is received at its Junction Node, the node performs weighted ECMP (Equal-Cost Multi-Path) flow-based forwarding across all egress SID lists associated with the Junction Segment.

As per [RFC9256] section 2.11, The fraction of the flows associated with a given segment list is w/Sw, where w is the weight of the segment list and Sw is the sum of the weights of the segment lists. To exclude forwarding via a specific egress interface while preserving the forwarding structure, a weight value of zero is assigned to the corresponding segment list.

7. Constraints

When the controller computes the DAG, traffic engineering constraints MUST be considered. Links which violate the constraints are pruned from the DAG. Nodes which do not form the DAG are not notified with any Junction segments.

8. Protection

As described in [I-D.draft-kompella-teas-mpte], as there are multiple egress interfaces (SID Lists), the loss of one interface link does not result in traffic drops, as long as one egress interface (SID List) remains although congestion may occur. For example, in Figure 1 Node C can tolerate the lost of up to 3 egress links and traffic will still forward (potentially with congestion).

If a Junction Node experiences a failure of all egress links (including any protection for those links), it will initially blackhole traffic until upstream nodes are notified by the controller to remove the failed Junction node from the DAG.

To reduce the risk of outages caused by single link failure, the controller MAY optimize DAG deployment by assigning Junction Segments only to nodes with more than one egress segment list. In other words, if a node in the DAG has only one egress interface, it functions solely as a transit node for an upstream Junction Segment and does not receive a Junction Segment itself. For example, in Figure 1, Node E is omitted from receiving a Junction Segment as it only has 1 egress link.

Link protection from an upstream Junction node to its downstream Junction nodes can be achieved using existing TI-LFA [I-D.draft-ietf-rtgwg-segment-routing-ti-lfa] mechanisms, applied per egress SID List on each Junction Segment. Since the top SID(s) in each SID List identify the path to the next downstream Junction node, TI-LFA is applicable. Effectively, TI-LFA is used to protect traffic between Junction Segments along a path within the DAG and is not intended to protect traffic directed toward the DAG' s egress nodes or the entire DAG.

Local computation for node protection on an upstream Junction node is not feasible, because it lacks visibility into the DAG beyond the immediate downstream Junction node as it only knows the next Junction Segment. A controller MAY be used to precompute backup SR Paths and signal these backup SID Lists to the upstream Junction segments.

9. Other considerations

9.1. Hierarchy

The use of Junction Segments to achieve a DAG can be used in hierarchical organization and sharing of DAGs between end-to-end tunnels.

For instance, in a multi-area or multi-instance topology, one or more shared DAGs may be created per area, connecting the border ingress node(s) and egress node(s). These DAGs can then be stitched together for use in an end-to-end SR tunnel.

Figure 2 is an example of two independent SR Policy Tunnels from Headend A and Headend B terminating on Egress X. An instance of a DAG with MID 100 can be configured between ABR-1 and the Egress X node. ABR-1 Junction Segment which ingresses the DAG has a Binding Segment attached, for example BSID-100. Therefore, the SID list on Headend A and Headend B SID lists would contain the SR Path (ex: Node-SID-ABR-1) to reach ABR 1 followed by BSID-100. The instruction set from each Headend to ABR 1 could also be another instance of a DAG, either for independent use or shared.

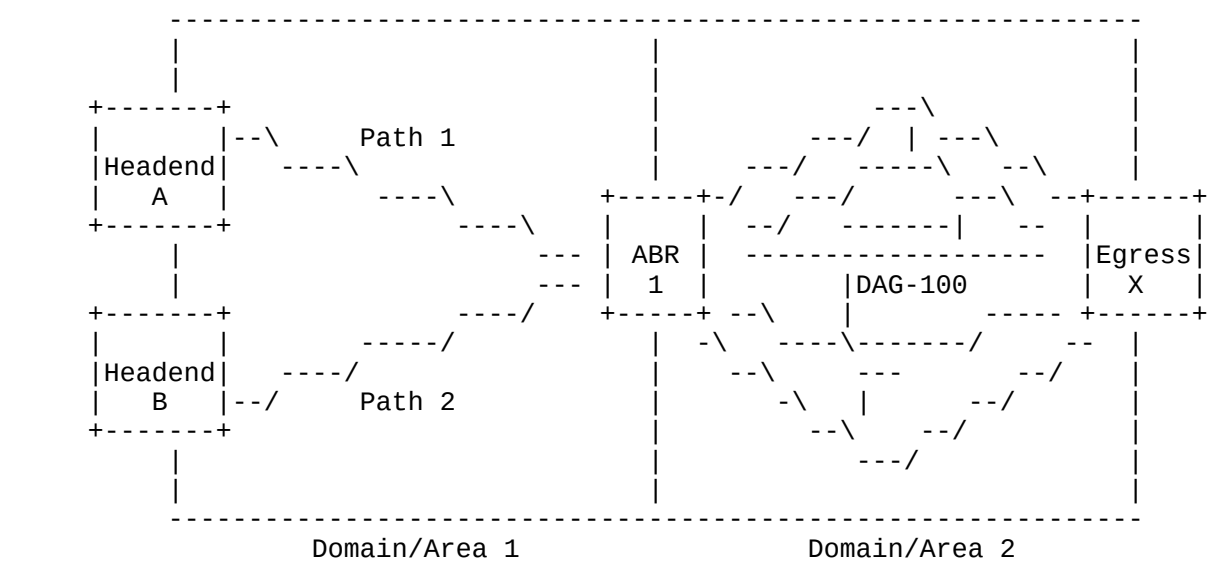

Figure 2 : Reusable DAG 100

9.2. Directly connected Junction nodes

As described in the Operational section, all transit nodes in a DAG MAY be signaled with a Junction Segment. Alternatively depending on the topological graph and TE requirements, two Junction segments may be interconnected via an SR Path, with a SID list, where the SR Path itself may correspond to an ECMP or TE Path.

9.3. Broadcast links

SR networks abstract broadcast links with the use point to point adjacency segments identifying each neighbor. Specifying a DAG which contains a broadcast link is feasible as an adjacency segment can be used to identify the neighboring Junction node on the broadcast link. The outgoing SID List of the Junction Segment simply contains the adjacency SID of the next-hop neighbor on the broadcast link.

10. Optimization

10.1. Local optimization

Local optimization refers to updates applied to a single Junction node that can be performed without requiring coordinated updates across multiple nodes in the DAG. These operations are considered safe when they do not introduce forwarding loops or cause reachability interruptions within the DAG.

Examples of local optimizations include:

* Manipulating the weight distribution of the outgoing SID lists

* Replacing an existing SID list with a more optimal one (e.g. using a new adjacency SID or updated SR path to the next Junction node).

* Adding a new SID list toward an egress node that is not itself a Junction node.

* Removing a SID list, provided that at least one other outgoing SID list remains active.

* Adding a new SID list to a Junction node that connects to a new downstream sub-graph which is not yet connected to the DAG structure.

While these operations are scoped to a single node, their correct application may still require awareness of the surrounding DAG structure to avoid introducing loops or reachability issues. For instance, the addition of a new SID list may require confirmation that the new sub-graph is not already part of the DAG, or that it connects loop free.

Local optimizations may also be chained in sequence across multiple nodes. When each step maintains loop free and reachable properties, such a chain of updates is still considered local in nature.

For example:

* Cleaning up a upstream disconnected sub-graph following the removal of an upstream SID list (BSID is no longer used)

It is important to note that local optimization differs from global optimization in that it does not require versioning or re-signaling of the entire DAG structure.

Since a Junction Segment is realized via an SR Policy, the controller leverages existing protocol mechanisms to update a Junction segment forwarding instructions, as the Junction Segment itself is represented as a candidate path within the SR Policy. When updating an existing candidate path, the binding SID MUST NOT change, as doing so would cause traffic using that binding SID to drop until upstream consumers are updated. Such a change should instead be treated as a global optimization, not a local one.

Multiple candidate paths could be used to support more comprehensive local optimizations. However, caution is required in how the MPTED version is encoded and how version increments are signaled, since candidate paths are signaled within the context of the SR Policy.

If the same color value is reused and a new candidate path is deployed, the new candidate path will not be installed in the forwarding plane until the existing one is removed, as only one candidate path may be active per SR Policy. This operation may not be hitless, depending on hardware implementation.

Alternatively, if a different color value is used, the new candidate path may be allowed to go active. However, it will still fail to do so since it should be using the same binding SID as the existing candidate path. This results in a collision, rendering the new path ineligible, and may also violate protocol constraints that prohibit such configurations.

Therefore, it is RECOMMENDED that local optimization be performed by updating the SID list of the existing candidate path, rather than introducing new candidate paths.

10.1.1. Local optimization Example

The following examples use the topology example specified in Figure 1 and section 5.1.

Example 1 Simple SID list update

* The Junction Segment B is updated. The SID list [Node-SID-H] is replaced with SID List [Adj-SID-BE, Adj-SID-EH]

Example 2 Chained Local Optimization

* The ingress SR Policy SID list is updated to remove { SID List 1: [Adj-SID-AB, BSID-B] }. Only SID List 2 and SID List 3 remain in the SR Policy candidate path

* The Junction Segment B is no longer used.

* A delete is sent to remove Junction Segment B

10.2. Global optimization

Global optimization of a DAG requires coordinated updates across all participating Junction Nodes. This process follows a make-before-break model, where a new version of the DAG is deployed alongside the existing one. The ingress SR Policy (or policies) is the last element to be updated.

Since the Color field of an SR Policy indicates DAG membership, and is managed by the controller, global optimization with make-before-break considerations necessitates deploying new Junction segments. This, in turn, requires the instantiation of new SR Policies with unique Color values. These new SR Policies are deployed to all Junction Nodes participating in the updated DAG instance and must be associated with distinct Binding SID values from those of the existing instance.

To minimize version churn and both color and label consumption, it is RECOMMENDED that controllers limit the number of concurrently deployed DAG versions for a given Multi-Point-to-Edge (MPTE) tunnel. Under normal conditions, only a single active DAG version should exist. During transitions, at most two versions (the current and the new one) should be present. Color and label values may be reused once globally released.

The Binding SID associated with a DAG instance MUST remain constant during local optimizations, but MUST change during global optimizations to avoid the risk of routing loops. Therefore, in the example above, if a Junction Node is participating in DAG #1, version #1 would be using one binding SID value and version #2 is using another.

Once all Junction segments are deployed to all Junction nodes, the ingress node is updated with new SID lists referencing the Binding SIDs of the new Junction segments downstream.

The Color field on ingress nodes is critical for traffic steering and therefore MUST remain constant during global optimizations. In contrast, Color values used on Junction Nodes MAY be reused or encoded to reflect DAG membership or instance mappings. Similarly, the binding SID on the ingress SR Policy also remains constant.

The controller tracks the Color values and their corresponding usage for DAG ID and version. For example:

* Color value 500 DAG #1 Version #1

* Color value 501 DAG #2 Version #1

* Color value 502 DAG #1 Version #2

When performing global updates, controllers SHOULD ensure that all Junction Nodes are updated in a coordinated and consistent manner before activating the new DAG on the ingress. If the update process fails partially, the controller SHOULD roll back the new deployment and retain the existing stable DAG version and it’s Junction Segments to avoid inconsistencies in forwarding behavior.

10.2.1. Global Optimization Example

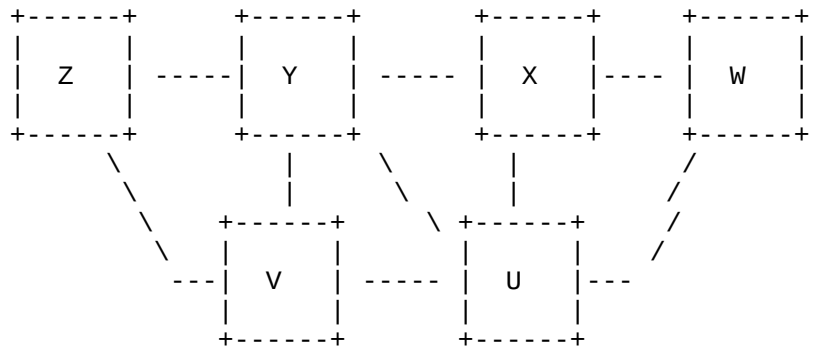

Figure 3 : Global MBB Example Topology

Step 0 - Initial State

Figure 3 gives an example topology containing a DAG which is to undergo optimization. Node Z is the ingress Node with an SR Policy color 1000 representing a service intent. Node W is the egress node. Color 2000 is the DAG MID and version tracked by the controller. In the existing state of the DAG, traffic flows from west to east, and north to south on nodes Y and X. The diagonal link between Y and U is currently not used.

```
Junction Segment Y1:
  Color: 2000
  BSID: BSID-Y1
  SID List 1: [Adj-SID-YX, BSID-X1]
  SID List 2: [Adj-SID-YV, Adj-SID-VU]

Junction Segment X1:
  Color: 2000
  BSID: BSID-X1
  SID List 1: [Adj-SID-XW]
  SID List 2: [Adj-SID-XU, Adj-SID-UW]

Ingress SR Policy:
  Color: 1000
  Candidate Path 1:
  SID List 1: [Adj-SID-ZY, BSID-Y1]
  SID List 2: [Adj-SID-ZV, Adj-SID-VU, Adj-SID-UW]
```

Step 1 - Deploy new Junction Segments

An optimization calculation occurs requiring the DAG to change, the traffic will now flow from south to north instead of north to south, and the diagonal link between Y and U will be used.

The controller assigns and tracks color 2001 for the new DAG. The following new Junction segments are deployed in the following order. The BSID values are either assigned by the controller, or requested by the node depending on the protocol in use. Note that after this deployment node Y has two Junction segments installed on it, both of which are active.

```
CREATE Junction Segment U2:
  Color: 2001
  BSID: BSID-U2
  SID List 1: [Adj-SID-UX, ADJ-SID-XW]
  SID List 2: [Adj-SID-UW]

CREATE Junction Segment Y2:
  Color: 2001
  BSID: BSID-Y2
  SID List 1: [Adj-SID-YX, ADJ-SID-XW]
  SID List 2: [Adj-SID-YU, BSID-U2]

CREATE Junction Segment V2:
  Color: 2001
  BSID: BSID-V2
  SID List 1: [Adj-SID-VY, BSID-Y2]
  SID List 2: [Adj-SID-VU, BSID-U2]
```

Step 2 - Update ingress nodes

The existing SR Policy candidate path is updated to use the new DAG. Note, the color and any associated binding SID values remain.

```
UPDATE Ingress SR Policy:
  Color: 1000
  Candidate Path 1:
    SID List 1: [Adj-SID-ZY, BSID-Y2]
    SID List 2: [Adj-SID-ZV, BSID-V2]
```

Step 3 - Delete the no longer used Junction segments

```
DELETE Junction Segment Y1:
  Color: 2000
  BSID: BSID-Y1

DELETE Junction Segment X1:
  Color: 2000
  BSID: BSID-X1
```

11. Security Considerations

TODO

12. Manageability Considerations

This document currently proposes using the existing SR Policy construct with a color value representing the DAG MID and version. Since SR Policy color value is originally intended for ingress traffic steering on matched routers, a deployment MUST allocate a color range which will be used for MPTE and MUST NOT be assigned to any advertised routes in the network.

TODO

13. IANA Considerations

None at this time

14. References

14.1. Normative References

[I-D.draft-ietf-idr-bgp-ls-sr-policy]
Previdi, S., Talaulikar, K., Dong, J., Gredler, H., and J. Tantsura, "Advertisement of Segment Routing Policies using BGP Link-State", Work in Progress, Internet-Draft, draft-

ietf-idr-bgp-ls-sr-policy-17, 6 March 2025,
<https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-ls-sr-policy-17>.

[I-D.draft-ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., and D. Jain, "Advertising Segment Routing Policies in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-segment-routing-te-policy-26, 23 October 2023,
<https://datatracker.ietf.org/doc/html/draft-ietf-idr-segment-routing-te-policy-26>.

[I-D.draft-ietf-rtgwg-segment-routing-ti-lfa]
Bashandy, A., Litkowski, S., Filsfils, C., Francois, P., Decraene, B., and D. Voyer, "Topology Independent Fast Reroute using Segment Routing", Work in Progress, Internet-Draft, draft-ietf-rtgwg-segment-routing-ti-lfa-21, 12 February 2025,
<https://datatracker.ietf.org/doc/html/draft-ietf-rtgwg-segment-routing-ti-lfa-21>.

[I-D.draft-kompella-teas-mpte]
Kompella, K., Jalil, L., Khaddam, M., and A. Smith, "Multipath Traffic Engineering", Work in Progress, Internet-Draft, draft-kompella-teas-mpte-01, 7 July 2025,
<https://datatracker.ietf.org/doc/html/draft-kompella-teas-mpte-01>.

[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/rfc/rfc2119>.

[RFC5440] Vasseur, JP., Ed. and JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", RFC 5440,
DOI 10.17487/RFC5440, March 2009,
<https://www.rfc-editor.org/rfc/rfc5440>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

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

[RFC8402] Filsfils, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <https://www.rfc-editor.org/rfc/rfc8402>.

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

14.2. Informative References

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

Acknowledgments

None at this time

Authors' Addresses

Andrew Stone
Nokia
Email: andrew.stone@nokia.com

Vishnu Pavan Beeram
Juniper Networks
Email: vbeeram@juniper.net

Nick Buraglio
Energy Sciences Network
Email: buraglio@forwardingplane.net

Shaofu Peng
ZTE Corporation
Email: peng.shaofu@zte.com.cn