BESS
Internet-Draft
Intended status: Standards Track
Expires: October 13, 2022

Z. Zhang
Juniper Networks
R. Raszuk
NTT Network Innovations
D. Pacella
Verizon
A. Gulko
Edward Jones Wealth Management

April 11, 2022

Controller Based BGP Multicast Signaling
draft-ietf-bess-bgp-multicast-controller-09

Abstract

This document specifies a way that one or more centralized controllers can use BGP to set up multicast distribution trees (identified by either IP source/destination address pair, mLDP FEC, or SR-P2MP Tree-ID) in a network. Since the controllers calculate the trees, they can use sophisticated algorithms and constraints to achieve traffic engineering. The controllers directly signal dynamic replication state to tree nodes, leading to very simple multicast control plane on the tree nodes, as if they were using static routes. This can be used for both underlay and overlay multicast trees, including replacing BGP-MVPN signaling.

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.

Status of This Memo

Copyright Notice

Table of Contents

1. Overview

1.1. Introduction

[I-D.ietf-bess-bgp-multicast] describes a way to use BGP as a replacement signaling for PIM [RFC7761] or mLDP [RFC6388]. The BGP-based multicast signaling described there provides a mechanism for setting up both (s,g)/(*,g) multicast trees (as PIM does, but optionally with labels) and labeled (MPLS) multicast tunnels (as mLDP does). Each router on a tree performs essentially the same procedures as it would perform if using PIM or mLDP, but all the inter-router signaling is done using BGP.

These procedures allow the routers to set up a separate tree for each individual multicast (x,g) flow where the 'x' could be either 's' or '*', but they also allow the routers to set up trees that are used for more than one flow. In the latter case, the trees are often referred to as "multicast tunnels" or "multipoint tunnels", and specifically in this document they are mLDP tunnels (except that they are set up with BGP signaling). While it actually does not have to be restricted to mLDP tunnels, mLDP FEC is conveniently borrowed to identify the tunnel. In the rest of the document, the term tree and tunnel are used interchangeably.

The trees/tunnels are set up using the "receiver-initiated join" technique of PIM/mLDP, hop by hop from downstream routers towards the root. The BGP messages of MCAST-TREE SAFI are either sent hop by hop between downstream routers and their upstream neighbors, or can be reflected by Route Reflectors (RRs).

As an alternative to each hop independently determining its upstream router and signaling upstream towards the root (following PIM/mLDP model), the entire tree can be calculated by a centralized controller, and the signaling can be entirely done from the

controller using the same MCAST-TREE SAFI. For that, some additional procedures and optimizations are specified in this document.

[I-D.ietf-bess-bgp-multicast] uses S-PMSI, Leaf, and Source Active Auto-Discovery (A-D) routes because the main procedures and concepts are borrowed from the BGP-MVPN [RFC6514]. While the same Leaf A-D routes can be used to signal replication state to tree nodes from controllers, this document introduces a new route type "Replication State" for the same functionality, so that familiarity with the BGP-MVPN concepts is not required.

While it is outside the scope of this document, signaling from the controllers could be done via other means as well, like Netconf or any other SDN methods.

1.2. Resilience

Each router could establish direct BGP sessions with one or more controllers, or it could establish BGP sessions with RRs who in turn peer with controllers. For the same tree/tunnel, each controller may independently calculate the tree/tunnel and signal the routers on the tree/tunnel using MCAST-TREE Replication State routes. How the calculation is done are outside the scope of this document.

On each router, BGP route selection rules will lead to one controller's route for the tree/tunnel being selected as the active route and used for setting up forwarding state. As long as all the routers on a tree/tunnel consistently pick the same controller's routes for the tree/tunnel, the setup should be consistent. If the tree/tunnel is labeled, different labels will be used from different controllers so there is no traffic loop issue even if the routers do not consistently select the same controlle's routes. In the unlabeled case, to ensure the consistency the selection SHOULD be solely based on the identifier of the controller.

Another consistency issue is when a bidirectional tree/tunnel needs to be re-routed. Because this is no longer triggered hop-by-hop from downstream to upstream, it is possible that the upstream change happens before the downstream, causing traffic loop. In the unlabeled case, there is no good solution (other than that the controller issues upstream change only after it gets acknowledgement from downstream). In the labeled case, as long as a new label is used there should be no problem.

Besides the traffic loop issue, there could be transient traffic loss before both the upstream and downstream's forwarding state are updated. This could be mitigated if the upstream keep sending traffic on the old path (in addition to the new path) and the

downstream keep accepting traffic on the old path (but not on the new path) for some time. It is a local matter when for the downstream to switch to the new path - it could be data driven (e.g., after traffic arrives on the new path) or timer driven.

For each tree, multiple disjoint instances could be calculated and signaled for live-live protection. Different labels are used for different instances, so that the leaves can differentiate incoming traffic on different instances. As far as transit routers are concerned, the instances are just independent. Note that the two instances are not expected to share common transit routers (it is otherwise outside the scope of this document/revision).

1.3. Signaling

When a router receives a Replication State route, the re-advertisement is blocked if a configured import RT matches the RT of the route, which indicates that this router is the target and consumer of the route hence it should not be re-advertised further. The routes includes the forwarding information in the form of Tunnel Encapsulation Attributes (TEA) [RFC9012], with enhancements specified in this document.

Suppose that for a particular tree, there are two downstream routers D1 and D2 for a particular upstream router U. A controller C sends one Replication State route to U, with the Tree Node's IP Address field (see Section 3.3) set to U's IP address and the TEA specifying both the two downstreams and its upstream (see Section 3.1.5). In this case, the Originating Router's Address field of the Replication State route is set to the controller's address. Note that for a TEA attached to a unicast NLRI, only one of the tunnels in a TEA is used for forwarding a particular packet, while all the tunnels in a TEA are used to reach multiple endpoints when it is attached to a multicast NLRI.

It could be that U may need to replicate to many downstream routers, say D1 through D1000. In that case, it may not be possible to encode all those branches in a single TEA, or may not be optimal to update a large TEA when a branch is added/removed. In that case, C may send multiple Replication State routes, each with a different Originating Router's Address field and a different TEA that encodes a subset of the branches. This provides a flexible way to optimize the encoding of large number of branches and incremental updates of branches.

Notice that, in case of labeled trees, the (x,g), mLDP FEC, or SR-P2MP tree identification (Section 1.7) signaling is actually not needed to transit routers but only needed to tunnel root/leaves. However, for consistency among the root/leaf/transit nodes, and for

consistency with the hop-by-hop signaling, the same signaling (with tree identification encoded in the NLRI) is used to all routers.

Nonetheless, a new NLRI route type of the MCAST-TREE SAFI is defined to encode label/SID instead of tree identification in the NLRI, for scenarios where there is really no need to signal tree identification, e.g. as described in Section 2. On a tunnel root, the tree's binding SID can be encoded in the NLRI.

For a tree node to acknowledge to the controller that it has received the signaling and installed corresponding forwarding state, it advertises a corresponding Replication State route, with the Originating Router's IP Address set to itself and with a Route Target to match the controller. For comparison, the tree signaling Replication State route from the controller has the Originating Router's IP Address set to the controller and the Route Target matching the tree node. The two Replication State routes (for controller to signal to a tree node and for a tree node to acknowledge back) differ only in those two aspects.

With the acknowledgement Replication State routes, the controller knows if tree setup is complete. The information can be used for many purposes, e.g. the controller may instruct the ingress to start forwarding traffic onto a tree only after it knows that the tree setup has completed.

1.4. Label Allocation

In the case of labeled multicast signaled hop by hop towards the root, whether it's (x,g) multicast or "mLDP" tunnel, labels are assigned by a downstream router and advertised to its upstream router (from traffic direction point of view). In the case of controller based signaling, routers do not originate tree join routes anymore, so the controllers have to assign labels on behalf of routers, and there are three options for label assignment:

- From each router's SRLB that the controller learns

- From the common SRGB that the controller learns

- From the controller's local label space

Assignment from each router's SRLB is no different from each router assigning labels from its own local label space in the hop-by-hop signaling case. The assignments for one router is independent of assignments for another router, even for the same tree.

Assignment from the controller's local label space is upstream-assigned [RFC5331]. It is used if the controller does not learn the common SRGB or each router's SRLB. Assignment from the SRGB [RFC8402] is only meaningful if all SRGBs are the same and a single common label is used for all the routers on a tree in case of unidirectional tree/tunnel (Section 1.4.1). Otherwise, assignment from SRLB is preferred.

The choice of which of the options to use depends on many factors. An operator may want to use a single common label per tree for ease of monitoring and debugging, but that requires explicit RPF checking and either common SRGB or upstream assigned labels, which may not be supported due to either the software or hardware limitations (e.g. label imposition/disposition limits). In an SR network, assignment from the common SRGB if it's required to use a single common label per unidirectional tree, or otherwise assignment from SRLB is a good choice because it does not require support for context label spaces.

1.4.1. Using a Common per-tree Label for All Routers

MPLS labels only have local significance. For an LSP that goes through a series of routers, each router allocates a label independently and it swaps the incoming label (that it advertised to its upstream) to an outgoing label (that it received from its downstream) when it forwards a labeled packet. Even if the incoming and outgoing labels happen to be the same on a particular router, that is just incidental.

With Segment Routing, it is becoming a common practice that all routers use the same SRGB so that a SID maps to the same label on all routers. This makes it easier for operators to monitor and debug their network. The same concept applies to multicast trees as well - a common per-tree label can be used for a router to receive traffic from its upstream neighbor and replicate traffic to all its downstream neighbor.

However, a common per-tree label can only be used for unidirectional trees. Additionally, unless the entire tree is updated for every tree node to use a new common per-tree label with any change in the tree (no matter how small and local the change is), it requires each router to do explicit RPF check, so that only packets from its expected upstream neighbor are accepted. Otherwise, traffic loop may form during topology changes, because the forwarding state update is no longer ordered.

Traditionally, p2mp mpls forwarding does not require explicit RPF check as a downstream router advertises a label only to its upstream router and all traffic with that incoming label is presumed to be

from the upstream router and accepted. When a downstream router switches to a different upstream router a different label will be advertised, so it can determine if traffic is from its expected upstream neighbor purely based on the label. Now with a single common label used for all routers on a tree to send and receive traffic with, a router can no longer determine if the traffic is from its expected neighbor just based on that common tree label. Therefore, explicit RPF check is needed. Instead of interface based RPF checking as in PIM case, neighbor based RPF checking is used - a label identifying the upstream neighbor precedes the common tree label and the receiving router checks if that preceding neighbor label matches its expected upstream neighbor. Notice that this is similar to what's described in Section "9.1.1 Discarding Packets from Wrong PE" of RFC 6513 (an egress PE discards traffic sent from a wrong ingress PE). The only difference is one is used for label based forwarding and the other is used for (s,g) based forwarding. [note: for bidirectional trees, we may be able to use two labels per tree - one for upstream traffic and one for downstream traffic. This needs further verification].

Both the common per-tree label and the neighbor label are allocated either from the common SRGB or from the controller's local label space. In the latter case, an additional label identifying the controller's label space is needed, as described in the following section.

1.4.2. Upstream-assignment from Controller's Local Label Space

In this case in the multicast packet's label stack the tree label and upstream neighbor label (if used in case of single common-label per tree) are preceded by a downstream-assigned "context label". The context label identifies a context-specific label space (the controller's local label space), and the upstream-assigned label that follows it is looked up in that space.

This specification requires that, in case of upstream-assignment from a controller's local label space, each router D to assign, corresponding to each controller C, a context label that identifies the upstream-assigned label space used by that controller. This label, call it Lc-D, is communicated by D to C via BGP-LS [RFC 7752].

Suppose a controller is setting up unidirectional tree T. It assigns that tree the label Lt, and assigns label Lu to identify router U which is the upstream of router D on tree T. C needs to tell U: "to send a packet on the given tree/tunnel, one of the things you have to do is push Lt onto the packet's label stack, then push Lu, then push Lc-D onto the packet's label stack, then unicast the packet to D".

Controller C also needs to inform router D of the correspondence between <Lc-D, Lu, Lt> and tree T.

To achieve that, when C sends a Replication State route, for each tunnel in the TEA, it may include a label stack Sub-TLV [RFC9012], with the outer label being the context label Lc-D (received by the controller from the corresponding downstream), the next label being the upstream neighbor label Lu, and the inner label being the label Lt assigned by the controller for the tree. The router receiving the route will use the label stacks to send traffic to its downstreams.

For C to signal the expected label stack for D to receive traffic with, we overload a tunnel TLV in the TEA of the Replication State route sent to D - if the tunnel TLV has a RPF sub-TLV (Section 3.1.5), then it indicates that this is actually for receiving traffic from the upstream.

1.5. Determining Root/Leaves

For the controller to calculate a tree, it needs to determine the root and leaves of the tree. This may be based on provisioning (static or dynamically programmed), or based on BGP signaling as described in the following two sections.

In both of the following cases, the BGP updates are targeted at the controller, via an address specific Route Target with Global Administration Field set to the controller's address and the Local Administration Field set to 0.

1.5.1. PIM-SSM/Bidir or mLDP

In this case, the PIM Last Hop Routers (LHRs) with interested receivers or mLDP tunnel leaves encode a Leaf A-D route ([I-D.ietf-bess-bgp-multicast]) with the Upstream Router's IP Address field set to the controller's address and the Originating Router's IP Address set to the address of the LHR or the P2MP tunnel leaf. The encoded PIM SSM source or mLDP FEC provides root information and the Originating Router's IP Address provides leaf information.

1.5.2. PIM ASM

In this case, the First Hop Routers (FHRs) originate Source Active routes which provides root information, and the LHRs originate Leaf A-D routes, encoded as in the PIM-SSM case except that it is (*,G) instead of (S,G). The Leaf A-D routes provide leaf information.

1.6. Multiple Domains

An end to end multicast tree may span multiple routing domains, and the setup of the tree in each domain may be done differently as specified in [I-D.ietf-bess-bgp-multicast]. This section discusses a few aspects specific to controller signaling.

Consider two adjacent domains each with its own controller in the following configuration where router B is an upstream node of C for a multicast tree:

```
                            |
                  domain 1  |  domain 2
                            |
                   ctrlr1   |   ctrlr2
                     /\     |     /\
                    /  \    |    /  \
                   /    \   |   /    \
                  A--...-B--|--C--...-D
                            |
```

In the case of native (un-labeled) IP multicast, nothing special is needed. Controller 1 signals B to send traffic out of B-C link while Controller 2 signals C to accept traffic on the B-C link.

In the case of labeled IP multicast or mLDP tunnel, the controllers may be able to coordinate their actions such that Controller 1 signals B to send traffic out of B-C link with label X while Controller 2 signals C to accept traffic with the same label X on the B-C link. If the coordination is not possible, then C needs to use hop-by-hop BGP signaling to signal towards B, as specified in [I-D.ietf-bess-bgp-multicast].

The configuration could also be as following, where router B borders both domain 1 and domain 2 and is controlled by both controllers:

```
                          |
                 domain 1 | domain 2
                          |
                   ctrlr1 | ctrlr2
                     /\   |   /\
                    /  \  |  /  \
                   /    \ | /    \
                  /      \|/      \
                 A--...---B--...---C
                          |
```

As discussed in Section 1.2, when B receives signaling from both Controller 1 and Controller 2, only one of the routes would be selected as the best route and used for programming the forwarding state of the corresponding segment. For B to stitch the two segments together, it is expected for B to know by provisioning that it is a border router so that B will look for the other segment (represented by the signaling from the other controller) and stitch the two together.

1.7. SR-P2MP

[I-D.ietf-pim-sr-p2mp-policy] describes an architecture to construct a Point-to-Multipoint (P2MP) tree to deliver Multi-point services in a Segment Routing domain. An SR P2MP tree is constructed by stitching together a set of Replication Segments that are specified in [I-D.ietf-spring-sr-replication-segment]. An SR Point-to-Multipoint (SR P2MP) Policy is used to define and instantiate a P2MP tree which is computed by a controller.

An SR P2MP tree is no different from an mLDP tunnel in MPLS forwarding plane. The difference is in control plane - instead of hop-by-hop mLDP signaling from leaves towards the root, to set up SR P2MP trees controllers program forwarding state (referred to as Replication Segments) to the root, leaves, and intermediate replication points using Netconf, PCEP, BGP or any other reasonable signaling/programming methods.

Procedures in this document can be used for controllers to set up SR P2MP trees with just an additional SR P2MP tree type and corresponding tree identification in the Replication State route.

If/once the SR Replication Segment is extended to bi-redirectional, and SR MP2MP is introduced, the same procedures in this document would apply to SR MP2MP as well.

2. Alternative to BGP-MVPN

Multicast with BGP signaling from controllers can be an alternative to BGP-MVPN [RFC6514]. It is an attractive option especially when the controller can easily determine the source and leaf information.

With BGP-MVPN, distributed signaling is used for the following:

o Egress PEs advertise C-multicast (Type-6/7) Auto-Discovery (A-D) routes to join C-multicast trees at the overlay (PE-PE).

- In case of ASM, ingress PEs advertise Source Active (Type-5) A-D routes to signal sources so that egress PEs can establish Shortest Path Trees (SPT).

- PEs advertise I/S-PMSI (Type-1/2/3) A-D routes to signal the binding of overlay/customer traffic to underlay/provider tunnels. For some types of tunnels, Leaf (Type-4) A-D routes are advertised by egress PEs in response to I/S-PMSI A-D routes to join the tunnels.

Based on the above signaled information, an ingress PE builds forwarding state to forward traffic arriving on the PE-CE interface to the provider tunnel (and local interfaces if there are local downstream receivers), and an egress PE builds forwarding state to forward traffic arriving on a provider tunnel to local interfaces with downstream receivers.

Notice that multicast with BGP signaling from controllers essentially programs "static" forwarding state onto multicast tree nodes. As long as a controller can determine how a C-multicast flow should be forwarded on ingress/egress PEs, it can signal to the ingress/egress PEs using the procedures in this document to set up forwarding state, removing the need of the above-mentioned distributed signaling and processing.

For the controller to learn the egress PEs for a C-multicast tree (so that it can set up or find a corresponding provider tunnel), the egress PEs advertise MCAST-TREE Leaf A-D routes (Section 1.5.1) towards the controller to signal its desire to join C-multicast trees, each with an appropriate RD and an extended community derived from the Route Target for the VPN ([I-D.zzhang-idr-rt-derived-community]) so that the controller knows which VPN it is for. The controller then advertises corresponding MCAST-TREE Replication State routes to set up C-multicast forwarding state on ingress and egress PEs. To encode the provider tunnel information in the MCAST-TREE Replication State route for an ingress PE, the TEA can explicitly list all replication branches of the tunnel, or just just the binding SID for the provider tunnel in the form of Segment List tunnel type, if the tunnel has a binding SID.

The Replication State route may also have a PMSI Tunnel Attribute (PTA) attached to specify the provider tunnel while the TEA specifies the local PE-CE interfaces where traffic need to be sent out. This not only allows provider tunnel without a binding SID (e.g., in a non-SR network) to be specified without explicitly listing its replication branches, but also allows the service controller for MVPN overlay state to be independent of provider tunnel setup (which could

be from a different transport controller or even without a controller).

However, notice that if the service controller and transport controller are different, then the service controller needs to signal the transport controller the tree information: identification, set of leaves, and applicable constraints. While this can be achieved (see Section 1.5.1), it is easier for the service and transport controller to be the same.

Depending on local policy, a PE may add PE-CE interfaces to its replication state based on local signaling (e.g., IGMP/PIM) instead of completely relying on signaling from controllers.

If dynamic switching between inclusive and selective tunnels based on data rate is needed, the ingress PE can advertise/withdraw S-PMSI routes targeted only at the controllers, without PMSI Tunnel Attribute attached. The controller then updates relevant MCAST-TREE Replication State routes to update C-multicast forwarding states on PEs to switch to a new tunnel.

3. Specification

3.1. Enhancements to TEA

A TEA may encode a list of tunnels. A TEA attached to an MCAST-TREE NLRI encodes replication information for a <tree, node > that is identified by the NRLI. Each tunnel in the TEA identifies a branch - either an upstream branch towards the tree root (Section 3.1.5) or a downstream branch towards some leaves. A tunnel in the TEA could have an outer encapsulation (e.g. MPLS label stack) or it could just be a one-hop direct connection for native IP multicast forwarding without any outer encapsulation.

This document specifies three new Tunnel Types and four new sub-TLVs. The type codes will be assigned by IANA from the "BGP Tunnel Encapsulation Attribute Tunnel Types".

3.1.1. Any-Encapsulation Tunnel

When a multicast packet needs to be sent from an upstream node to a downstream node, it may not matter how it is sent - natively when the two nodes are directly connected or tunneled otherwise. In case of tunneling, it may not matter what kind of tunnel is used - MPLS, GRE, IPinIP, or whatever.

To support this, an "Any-Encapsulation" tunnel type of value 20 is defined. This tunnel MAY have a Tunnel Egress Endpoint and other

Sub-TLVs. The Tunnel Egress Endpoint Sub-TLV specifies an IP address, which could be any of the following:

- An interface's local address - when a packet needs to sent out of the corresponding interface natively. On a LAN multicast MAC address MUST be used.

- A directly connected neighbor's interface address - when a packet needs to unicast to the address natively.

- An address that is not directly connected - when a packet needs to be tunneled to the address (any tunnel type/instance can be used).

3.1.2. Load-balancing Tunnel

Consider that a multicast packet needs to be sent to a downstream node, which could be reached via four paths P1~P4. If it does not matter which of path is taken, an "Any-Encapsulation" tunnel with the Tunnel Egress Endpoint Sub-TLV specifying the downstream node's loopback address works well. If the controller wants to specify that only P1~P2 should be used, then a "Load-balancing" tunnel needs to be used, listing P1 and P2 as member tunnels of the "Load-balancing" tunnel.

A load-balancing tunnel has one "Member Tunnels" Sub-TLV defined in this document. The Sub-TLV is a list of tunnels, each specifying a way to reach the downstream. A packet will be sent out of one of the tunnels listed in the Member Tunnels Sub-TLV of the load-balancing tunnel.

3.1.3. Segment List Tunnel

A Segment List tunnel has a Segment List sub-TLV. The encoding of the sub-TLV is as specified in Section 2.4.4 of [I-D.ietf-idr-segment-routing-te-policy]. An example use of a Segment List tunnel is provided in Section 3.4.3.

3.1.4. Receiving MPLS Label Stack

While [I-D.ietf-bess-bgp-multicast] uses S-PMSI A-D routes to signal forwarding information for MP2MP upstream traffic, when controller signaling is used, a single Replication State route is used for both upstream and downstream traffic. Since different upstream and downstream labels need to be used, a new "Receiving MPLS Label Stack" of type TBD is added as a tunnel sub-TLV in addition to the existing MPLS Label Stack sub-TLV. Other than type difference, the two are the encoded the same way.

The Receiving MPLS Label Stack sub-TLV is added to each downstream tunnel in the TEA of Replication State route for an MP2MP tunnel to specify the forwarding information for upstream traffic from the corresponding downstream node. A label stack instead of a single label is used because of the need for neighbor based RPF check, as further explained in the following section.

The Receiving MPLS Label Stack sub-TLV is also used for downstream traffic from the upstream for both P2MP and MP2MP, as specified below.

3.1.5. RPF Sub-TLV

The RPF sub-TLV is of type 124 allocated by IANA and has a one-octet length. The length is 0 currently, but if necessary in the future, sub-sub-TLVs could be placed in its value part. If the RPF sub-TLV appears in a tunnel, it indicates that the "tunnel" is for the upstream node instead of a downstream node.

In case of MPLS, the tunnel contains an Receiving MPLS Label Stack sub-TLV for downstream traffic from the upstream node, and in case of MP2MP it also contains a regular MPLS Label Stack sub-TLV for upstream traffic to the upstream node.

The inner most label in the Receiving MPLS Label Stack is the incoming label identifying the tree (for comparison the inner most label for a regular MPLS Label Stack is the outgoing label). If the Receiving MPLS Label Stack sub-TLVe has more than one labels, the second inner most label in the stack identifies the expected upstream neighbor and explicit RPF checking needs to be set up for the tree label accordingly.

3.1.6. Tree Label Stack sub-TLV

The MPLS Label Stack sub-TLV can be used to specify the complete label stack used to send traffic, with the stack including both a transport label (stack) and label(s) that identify the (tree, neighbor) to the downstream node. There are cases where the controller only wants to specify the tree-identifying labels but leave the transport details to the router itself. For example, the router could locally determine a transport label (stack) and combine with the tree-identifying labels signaled from the controller to get the complete outgoing label stack.

For that purpose, a new Tree Label Stack sub-TLV of type 125 is defined, with a one-octet length field. It MAY appear in an Any-Encapsulation tunnel. The value field contains a label stack with the same encoding as value part of the MPLS Label Stack sub-TLV, but

with a different type. A stack is specified because it may take up to three labels (see Section 1.4):

- o If different nodes use different labels (allocated from the common SRGB or the node's SRLB) for a (tree, neighbor) tuple, only a single label is in the stack. This is similar to current mLDP hop by hop signaling case.

- o If different nodes use the same tree label, then an additional neighbor-identifying label is needed in front of the tree label.

- o For the previous bullet, if the neighbor-identifying label is allocated from the controller's local label space, then an additional context label is needed in front of the neighbor label.

3.1.7. Backup Tunnel sub-TLV

The Backup Tunnel sub-TLV is used to specify the backup paths for an Any-Encapsulation or Segment List tunnel. The length is two-octet. The value part encodes a one-octet flags field and a variable length Tunnel Encapsulation Attribute. If the tunnel goes down, traffic that is normally sent out of the tunnel is fast rerouted to the tunnels listed in the encoded TEA.

```
               +---------------------------------+
               | Sub-TLV Type (1 Octet, TBD)     |
               +---------------------------------+
               | Sub-TLV Length (2 Octets)       |
               +---------------------------------+
               | P | rest of 1 Octet Flags       |
               +---------------------------------+
               | Backup TEA (variable length)    |
               +---------------------------------+
```

The backup tunnels can be going to the same or different nodes reached by the original tunnel.

If the tunnel carries a RPF sub-TLV and a Backup Tunnel sub-TLV, then both traffic arriving on the original tunnel and on the tunnels encoded in the Backup Tunnel sub-TLV's TEA can be accepted, if the Parallel (P-)bit in the flags field is set. If the P-bit is not set, then traffic arriving on the backup tunnel is accepted only if router has switched to receiving on the backup tunnel (this is the equivalent of PIM/mLDP MoFRR).

3.2. Context Label TLV in BGP-LS Node Attribute

For a router to signal the context label that it assigns for a controller (or any label allocator that assigns labels - from its local label space - that will be received by this router), a new BGP-LS Node Attribute TLV is defined:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |              Type             |            Length             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Context Label                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            IPv4/v6 Address of Label Space Owner               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Length field implies the type of the address. Multiple Context Label TLVs may be included in a Node Attribute, one for each label space owner.

An as example, a controller with address 11.11.11.11 allocates label 200 from its own label space, and router A assigns label 100 to identify this controller's label space. The router includes the Context Label TLV (100, 11.11.11.11) in its BGP-LS Node Attribute and the controller instructs router B to send traffic to router A with a label stack (100, 200), and router A uses label 100 to determine the Label FIB in which to look up label 200.

3.3. Replicate State Route Type

The NLRI route type for signaling from controllers to tree nodes is "Replication State". The NLRI has the following format:

```
                   +-----------------------------------+
                   |  Route Type - Replication State   |
                   +-----------------------------------+
                   |     Length (1 octet)              |
                   +-----------------------------------+
                   |     Tree Type (1 octet)           |
                   +-----------------------------------+
                   |Tree Type Specific Length (1 octet)|
                   +-----------------------------------+
                   ~  Tree Identification (variable)   ~
                   +-----------------------------------+
                   |    Tree Node's IP Address         |
                   +-----------------------------------+
                   |    Originator's IP Address        |
                   +-----------------------------------+
```

Replication State NLRI

Notice that Replication State is just a new route type with the same format of Leaf A-D route except some fields are renamed:

- Tree Type in Replication State route matches the PMSI route type in Leaf A-D route

- Tree Node's IP Address matches the Upstream Router's IP Address of the PMSI route key in Leaf A-D route

With this arrangement, IP multicast tree and mLDP tunnel can be signaled via Replication State routes from controllers, or via Leaf A-D routes either hop by hop or from controllers with maximum code reuse, while newer types of trees like SR-P2MP can be signaled via Replication State routes with maximum code reuse as well.

3.4. SR P2MP Signaling

An SR P2MP policy for an SR P2MP tree is identified by a (Root, Tree-id) tuple. It has a set of leaves and set of Candidate Paths (CPs). The policy is instantiated on the root of the tree, with corresponding Replication Segments - identified by (Root, Tree-id, Tree-Node-id) - instantiated on the tree nodes (root, leaves, and intermediate replication points).

3.4.1. Replication State Route for SR P2MP

For SR P2MP, forwarding on tree nodes state are represented as Replication Segments and are signaled from controllers to tree nodes via Replication State routes. A Replication State route for SR P2MP has a Tree Type 1 and the Tree Identification includes (Route

Distinguisher, Root ID, Tree ID), where the RD implicitly identifies the candidate path.

```
                   +-----------------------------------+
                   |   Route Type - Replication State  |
                   +-----------------------------------+
                   |     Length (1 octet)              |
                   +-----------------------------------+
                   |    Tree Type (1 - SR P2MP)        |
                   +-----------------------------------+
                   |Tree Type Specific Length (1 octet)|
                   +-----------------------------------+
                   |      RD   (8 octets)              |
                   +-----------------------------------+
                   |  Root ID (4 or 16 octets)         |
                   +-----------------------------------+
                   |       Tree ID (4 octets)          |
                   +-----------------------------------+
                   |     Tree Node's IP Address        |
                   +-----------------------------------+
                   |  Originating Router's IP Address  |
                   +-----------------------------------+
```

Replication State route for SR Replication Segment

3.4.2. BGP Community Container for SR P2MP Policy

The Replication State route for Replication Segments signaled to the root is also used to signal (parts of) the SR P2MP Policy - the policy name, the set of leaves (optional, for informational purpose), preference of the CP and other information are all encoded in a newly defined BGP Community Container (BCC) [I-D.ietf-idr-wide-bgp-communities] called SR P2MP Policy BCC.

The SR P2MP Policy BCC has a BGP Community Container type to be assigned by IANA. It is composed of a fixed 4-octet Candidate Path Preference value, optionally followed by TLVs.

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                Candidate Path Preference                      |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                                                               |
       |                        TLVs (optional)                        |
       |                                                               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

BGP Community Container for SR P2MP Policy

One optional TLV is to enclose the following optional Atoms TLVs that are already defined in [I-D.ietf-idr-wide-bgp-communities]:

o An IPv4 or IPv6 Prefix list - for the set of leaves

o A UTF-8 string - for the policy name

If more information for the policy are needed, more Atoms TLVs or SR P2MP Policy BCC specific TLVs can be defined.

The root receives one Replication State route for each Candidate Path of the policy. Only one of the routes need to, though more than one MAY include the above listed optional Atom TLVs in the SR P2MP Policy BCC.

Alternatively, an additional route type can be used to carry policy information instead. Details/decision to be specified in a future revision.

3.4.3. Tunnel Encapsulation Attribute

The TEA attached to a Replication State route for SR-P2MP encodes tunnels as specified in earlier sections. A tunnel could be an Any-Encapsulation tunnel with MPLS Label Stack sub-TLV or Receiving MPLS Label Stack sub-TLV (in case of SR-MPLS), a Segment List tunnel, or a Load-balancing tunnel.

For a Segment List tunnel in this context, the last segment in the segment list represents the SID of the tree. When it is without the RPF sub-TLV, the previous segments in the list steer traffic to the downstream node, and the segment before the last one MAY also be a binding SID for another P2MP tunnel, meaning that the replication branch represented by this "Segment List" is actually a P2MP tunnel to a set of downstream nodes.

3.5. Replication State Route with Label Stack for Tree Identification

As described in Section 1.3, tree label instead of tree identification could be encoded in the NLRI to identify the tree in the control plane as well as in the forwarding plane. For that a new Tree Type of 2 is used and the Replication State route has the following format:

```
                 +-------------------------------------+
                 |    Route Type - Replication State   |
                 +-------------------------------------+
                 |     Length (1 octet)                |
                 +-------------------------------------+
                 |    Tree Type 2 (Label as Tree ID)   |
                 +-------------------------------------+
                 |Tree Type specific Length (1 octet)  |
                 +-------------------------------------+
                 |      RD   (8 octets)                |
                 +-------------------------------------+
                 ~      Label Stack (variable)         ~
                 +-------------------------------------+
                 |  Tree Node's IP Address             |
                 +-------------------------------------+
                 |  Originating Router's IP Address    |
                 +-------------------------------------+
```

Replication State route for tree identification by label stack

As discussed in Section 1.4.2, a label stack may have to be used to identify a tree in the data plane so a label stack is encoded here. The number of labels is derived from the Tree Type Specific Length field. Each label stack entry is encoded as following:

```
        0                   1                   2                   3
        0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                Label                  |0 0 0 0 0 0 0 0 0 0 0 0|
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4. Procedures

Details to be added. The general idea is described in the introduction section.

5. Security Considerations

This document does not introduce new security risks.

6. IANA Considerations

IANA has assigned the following code points:

- "Any-Encapsulation" tunnel type 78 from "BGP Tunnel Encapsulation Attribute Tunnel Types" registry

- "RPF" sub-TLV type 124 and "Tree Label Stack" sub-TLV type 125 from "BGP Tunnel Encapsulation Attribute Sub-TLVs" registry

This document makes the following additional IANA requests:

- Assign "Segment List" and "Load-balancing" tunnel types from the "BGP Tunnel Encapsulation Attribute Tunnel Types" registry

- Assign "Member Tunnels" and "Receiving MPLS Label Stack" sub-TLV types from the "BGP Tunnel Encapsulation Attribute Sub-TLVs" registry. The "Member Tunnels" sub-TLV has a two-octet value length (so the type should be in the 128-255 range), while the "Receiving MPLS Label Stack" sub-TLV has a one-octet value length.

- Assign "Context Label TLV" type from the "BGP-LS Node Descriptor, Link Descriptor, Prefix Descriptor, and Attribute TLVs" registry.

- Assign "Replication State" route type from the "BGP MCAST-TREE Route Types" registry.

- Create a "Tree Type Registry for Replication State Route", with the following initial assignments:

    * 1: SR-P2MP

    * 2: P2MP Tree with Label as Identification

    * 3: IP Multicast

    * 0x43: mLDP

- Assign a new BGP Community Container type "SR P2MP Policy", and to create an "SR P2MP Policy Community Container TLV Registry", with an initial entry for "TLV for Atoms".

7. Acknowledgements

The authors Eric Rosen for his questions, suggestions, and help finding solutions to some issues like the neighbor based explicit RPF checking. The authors also thank Lenny Giuliano, Sanoj Vivekanandan and IJsbrand Wijnands for their review and comments.

8. References

8.1. Normative References

[I-D.ietf-bess-bgp-multicast]
Zhang, Z., Giuliano, L., Patel, K., Wijnands, I., Mishra, M., and A. Gulko, "BGP Based Multicast", draft-ietf-bess-bgp-multicast-04 (work in progress), January 2022.

[I-D.ietf-idr-segment-routing-te-policy]
Previdi, S., Filsfils, C., Talaulikar, K., Mattes, P., Jain, D., and S. Lin, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-16 (work in progress), March 2022.

[I-D.ietf-idr-wide-bgp-communities]
Raszuk, R., Haas, J., Lange, A., Decraene, B., Amante, S., and P. Jakma, "BGP Community Container Attribute", draft-ietf-idr-wide-bgp-communities-06 (work in progress), January 2022.

[I-D.ietf-pim-sr-p2mp-policy]
(editor), D. V., Filsfils, C., Parekh, R., Bidgoli, H., and Z. Zhang, "Segment Routing Point-to-Multipoint Policy", draft-ietf-pim-sr-p2mp-policy-04 (work in progress), March 2022.

[I-D.ietf-spring-sr-replication-segment]
(editor), D. V., Filsfils, C., Parekh, R., Bidgoli, H., and Z. Zhang, "SR Replication Segment for Multi-point Service Delivery", draft-ietf-spring-sr-replication-segment-07 (work in progress), March 2022.

[I-D.zzhang-idr-rt-derived-community]
Zhang, Z., Haas, J., and K. Patel, "Extended Communities Derived from Route Targets", draft-zzhang-idr-rt-derived-community-02 (work in progress), March 2022.

[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>.

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

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

[RFC9012] Patel, K., Van de Velde, G., Sangli, S., and J. Scudder, "The BGP Tunnel Encapsulation Attribute", RFC 9012, DOI 10.17487/RFC9012, April 2021, <https://www.rfc-editor.org/info/rfc9012>.

8.2. Informative References

[RFC6388] Wijnands, IJ., Ed., Minei, I., Ed., Kompella, K., and B. Thomas, "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6388, DOI 10.17487/RFC6388, November 2011, <https://www.rfc-editor.org/info/rfc6388>.

[RFC6513] Rosen, E., Ed. and R. Aggarwal, Ed., "Multicast in MPLS/BGP IP VPNs", RFC 6513, DOI 10.17487/RFC6513, February 2012, <https://www.rfc-editor.org/info/rfc6513>.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <https://www.rfc-editor.org/info/rfc6514>.

[RFC7060] Napierala, M., Rosen, E., and IJ. Wijnands, "Using LDP Multipoint Extensions on Targeted LDP Sessions", RFC 7060, DOI 10.17487/RFC7060, November 2013, <https://www.rfc-editor.org/info/rfc7060>.

[RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <https://www.rfc-editor.org/info/rfc7761>.

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

Authors' Addresses

Zhaohui Zhang
Juniper Networks

EMail: zzhang@juniper.net

Robert Raszuk
NTT Network Innovations

EMail: robert@raszuk.net

Dante Pacella
Verizon

EMail: dante.j.pacella@verizon.com

Arkadiy Gulko
Edward Jones Wealth Management

EMail: arkadiy.gulko@edwardjones.com