BESS Workgroup
Internet Draft
Intended status: Standards Track

J. Rabadan, Ed.
S. Sathappan
W. Henderickx
Nokia

R. Shekhar
N. Sheth
W. Lin
M. Katiyar
Juniper

A. Sajassi
Cisco

A. Isaac
Juniper

M. Tufail
Citibank

Expires: August 19, 2017

February 15, 2017

Optimized Ingress Replication solution for EVPN
draft-ietf-bess-evpn-optimized-ir-01

Abstract

Network Virtualization Overlay (NVO) networks using EVPN as control plane may use ingress replication (IR) or PIM-based trees to convey the overlay BUM traffic. PIM provides an efficient solution to avoid sending multiple copies of the same packet over the same physical link, however it may not always be deployed in the NVO core network. IR avoids the dependency on PIM in the NVO network core. While IR provides a simple multicast transport, some NVO networks with demanding multicast applications require a more efficient solution without PIM in the core. This document describes a solution to optimize the efficiency of IR in NVO networks.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

This Internet-Draft will expire on August 19, 2017.

Table of Contents

1. Problem Statement 3
2. Solution requirements 4
3. EVPN BGP Attributes for optimized-IR 5
4. Non-selective Assisted-Replication (AR) Solution Description . 7
4.1. Non-selective AR-REPLICATOR procedures 8
4.2. Non-selective AR-LEAF procedures 9
4.3. RNVE procedures 11
4.4. Forwarding behavior in non-selective AR EVIs 11
4.4.1. Broadcast and Multicast forwarding behavior 11
4.4.1.1. Non-selective AR-REPLICATOR BM forwarding 11
4.4.1.2. Non-selective AR-LEAF BM forwarding 12
4.4.1.3. RNVE BM forwarding 12
4.4.2. Unknown unicast forwarding behavior 13
4.4.2.1. Non-selective AR-REPLICATOR/LEAF Unknown unicast forwarding 13
4.4.2.2. RNVE Unknown unicast forwarding 13
5. Selective Assisted-Replication (AR) Solution Description . . . 13
5.1. Selective AR-REPLICATOR procedures 14
5.2. Selective AR-LEAF procedures 15

5.3. Forwarding behavior in selective AR EVIs 16
5.3.1. Selective AR-REPLICATOR BM forwarding 16
5.3.2. Selective AR-LEAF BM forwarding 17
6. Pruned-Flood-Lists (PFL) 18
6.1. A PFL example 18
7. AR Procedures for single-IP AR-REPLICATORS 19
8. AR Procedures and EVPN Multi-homing Split-Horizon 20
9. Out-of-band distribution of Broadcast/Multicast traffic 21
10. Benefits of the optimized-IR solution 21
11. Conventions used in this document 21
12. Security Considerations 21
13. IANA Considerations 22
14. Terminology 22
15. References 23
15.1 Normative References 23
15.2 Informative References 23
16. Acknowledgments 23
17. Authors' Addresses 23

1. Problem Statement

EVPN may be used as the control plane for a Network Virtualization Overlay (NVO) network. Network Virtualization Edge (NVE) devices and PEs that are part of the same EVI use Ingress Replication (IR) or PIM-based trees to transport the tenant's BUM traffic. In NVO networks where PIM-based trees cannot be used, IR is the only alternative. Examples of these situations are NVO networks where the core nodes don't support PIM or the network operator does not want to run PIM in the core.

In some use-cases, the amount of replication for BUM (Broadcast, Unknown unicast and Multicast traffic) is kept under control on the NVEs due to the following fairly common assumptions:

a) Broadcast is greatly reduced due to the proxy-ARP and proxy-ND capabilities supported by EVPN on the NVEs. Some NVEs can even provide DHCP-server functions for the attached Tenant Systems (TS) reducing the broadcast even further.

b) Unknown unicast traffic is greatly reduced in virtualized NVO networks where all the MAC and IP addresses are learnt in the control plane.

c) Multicast applications are not used.

If the above assumptions are true for a given NVO network, then IR

provides a simple solution for multi-destination traffic. However, the statement c) above is not always true and multicast applications are required in many use-cases.

When the multicast sources are attached to NVEs residing in hypervisors or low-performance-replication TORs, the ingress replication of a large amount of multicast traffic to a significant number of remote NVEs/PEs can seriously degrade the performance of the NVE and impact the application.

This document describes a solution that makes use of two IR optimizations:

i) Assisted-Replication (AR)
ii) Pruned-Flood-Lists (PFL)

Both optimizations may be used together or independently so that the performance and efficiency of the network to transport multicast can be improved. Both solutions require some extensions to [RFC7432] that are described in section 3.

Section 2 lists the requirements of the combined optimized-IR solution, whereas sections 4 and 5 describe the Assisted-Replication (AR) solution, and section 6 the Pruned-Flood-Lists (PFL) solution.

2. Solution requirements

The IR optimization solution (optimized-IR hereafter) MUST meet the following requirements:

a) The solution MUST provide an IR optimization for BM (Broadcast and Multicast) traffic, while preserving the packet order for unicast applications, i.e. known and unknown unicast traffic SHALL follow the same path.

b) The solution MUST be compatible with [RFC7432] and [EVPN-OVERLAY] and have no impact on the EVPN procedures for BM traffic. In particular, the solution SHOULD support the following EVPN functions:

   - All-active multi-homing, including the split-horizon and Designated Forwarder (DF) functions.

   - Single-active multi-homing, including the DF function.

   - Handling of multi-destination traffic and processing of broadcast and multicast as per [RFC7432].

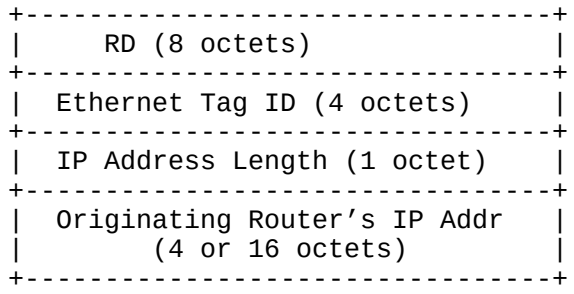
c) The solution MUST be backwards compatible with existing NVEs using a non-optimized version of IR. A given EVI can have NVEs/PEs supporting regular-IR and optimized-IR.

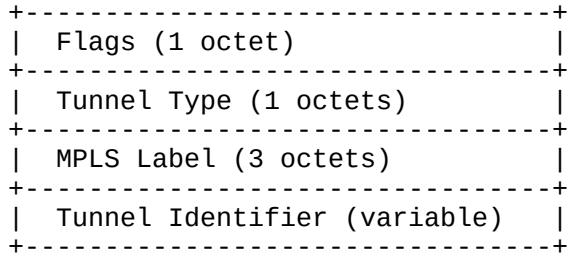
d) The solution MUST be independent of the NVO specific data plane encapsulation and the virtual identifiers being used, e.g.: VXLAN VNIs, NVGRE VSIDs or MPLS labels.

3. EVPN BGP Attributes for optimized-IR

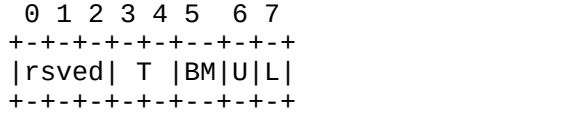
This solution proposes some changes to the [RFC7432] Inclusive Multicast Ethernet Tag routes and attributes so that an NVE/PE can signal its optimized-IR capabilities.

The Inclusive Multicast Ethernet Tag route (RT-3) and its PMSI Tunnel Attribute's (PTA) general format used in [RFC7432] are shown below:





The Flags field is defined as follows:



Where a new type field (for AR) and two new flags (for PFL signaling) are defined:

- T is the AR Type field (2 bits) that defines the AR role of the advertising router:

  + 00 (decimal 0) = RNVE (non-AR support)

  + 01 (decimal 1) = AR-REPLICATOR

  + 10 (decimal 2) = AR-LEAF

  + 11 (decimal 3) = RESERVED

- The PFL (Pruned-Flood-Lists) flags defined the desired behavior of the advertising router for the different types of traffic:

  + BM= Broadcast and Multicast (BM) flag. BM=1 means "prune-me" from the BM flooding list. BM=0 means regular behavior.

  + U= Unknown flag. U=1 means "prune-me" from the Unknown flooding list. U=0 means regular behavior.

- Flag L is an existing flag defined in [RFC6514] (L=Leaf Information Required) and it will be used only in the Selective AR Solution.

Please refer to section 10 for the IANA considerations related to the PTA flags.

In this document, the above RT-3 and PTA can be used in two different modes for the same EVI/Ethernet Tag:

o Regular-IR route: in this route, Originating Router’s IP Address, Tunnel Type (0x06), MPLS Label, Tunnel Identifier and Flags MUST be used as described in [RFC7432]. The Originating Router’s IP Address and Tunnel Identifier are set to an IP address that we denominate IR-IP in this document.

o Replicator-AR route: this route is used by the AR-REPLICATOR to advertise its AR capabilities, with the fields set as follows.

  + Originating Router’s IP Address as well as the Tunnel Identifier are set to the same routable IP address that we denominate AR-IP and SHOULD be different than the IR-IP for a given PE/NVE.

  + Tunnel Type = Assisted-Replication (AR). Section 11 provides the allocated type value.

  + T (AR role type) = 01 (AR-REPLICATOR).

  + L (Leaf Information Required) = 0 (for non-selective AR) or 1

(for selective AR).

In addition, this document also uses the Leaf-AD route (RT-11) defined in [EVPN-BUM] in case the selective AR mode is used. The Leaf-AD route MAY be used by the AR-LEAF in response to a Replicator-AR route (with the L flag set) to advertise its desire to receive the multicast traffic from a specific AR-REPLICATOR. It is only used for selective AR and its fields are set as follows:

+ Originating Router’s IP Address is set to the advertising IR-IP (same IP used by the AR-LEAF in regular-IR routes).

+ Route Key is the "Route Type Specific" NLRI of the Replicator-AR route for which this Leaf-AD route is generated.

+ The AR-LEAF constructs an IP-address-specific route-target as indicated in [EVPN-BUM], by placing the IP address carried in the Next Hop field of the received Replicator-AR route in the Global Administrator field of the Community, with the Local Administrator field of this Community set to 0. Note that the same IP-address-specific import route-target is auto-configured by the AR-REPLICATOR that sent the Replicator-AR, in order to control the acceptance of the Leaf-AD routes.

+ The leaf-AD route MUST include the PMSI Tunnel attribute with the Tunnel Type set to AR, type set to AR-LEAF and the Tunnel Identifier set to the IR-IP of the advertising AR-LEAF. The PMSI Tunnel attribute MUST carry a downstream-assigned MPLS label that is used by the AR-REPLICATOR to send traffic to the AR-LEAF.

Each AR-enabled node MUST understand and process the AR type field in the PTA (Flags field) of the routes, and MUST signal the corresponding type (1 or 2) according to its administrative choice.

Each node, part of the EVI, MAY understand and process the BM/U flags. Note that these BM/U flags may be used to optimize the delivery of multi-destination traffic and its use SHOULD be an administrative choice, and independent of the AR role.

Non-optimized-IR nodes will be unaware of the new PMSI attribute flag definition as well as the new Tunnel Type (AR), i.e. they will ignore the information contained in the flags field for any RT-3 and will ignore the RT-3 routes with an unknown Tunnel Type (type AR in this case).

4. Non-selective Assisted-Replication (AR) Solution Description

The following figure illustrates an example NVO network where the non-selective AR function is enabled. Three different roles are defined for a given EVI: AR-REPLICATOR, AR-LEAF and RNVE (Regular NVE). The solution is called "non-selective" because the chosen AR-REPLICATOR for a given flow MUST replicate the multicast traffic to 'all' the NVE/PEs in the EVI except for the source NVE/PE.

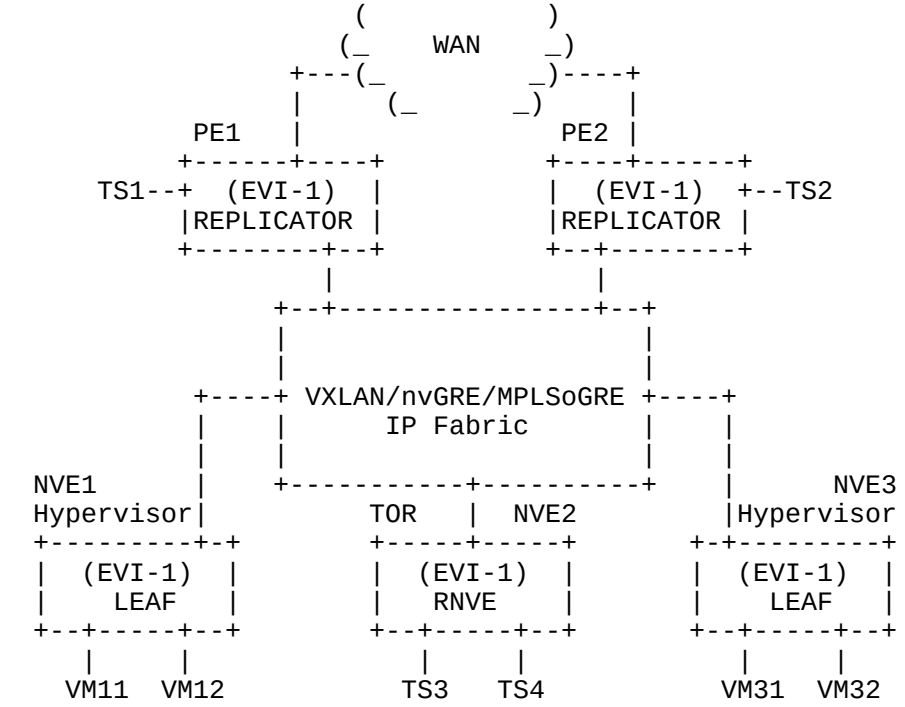


Figure 1 Optimized-IR scenario

4.1. Non-selective AR-REPLICATOR procedures

An AR-REPLICATOR is defined as an NVE/PE capable of replicating ingress BM (Broadcast and Multicast) traffic received on an overlay tunnel to other overlay tunnels and local Attachment Circuits (ACs). The AR-REPLICATOR signals its role in the control plane and understands where the other roles (AR-LEAF nodes, RNVEs and other AR-REPLICATORs) are located. A given AR-enabled EVI service may have zero, one or more AR-REPLICATORs. In our example in figure 1, PE1 and PE2 are defined as AR-REPLICATORs. The following considerations apply to the AR-REPLICATOR role:

a) The AR-REPLICATOR role SHOULD be an administrative choice in any

NVE/PE that is part of an AR-enabled EVI. This administrative option to enable AR-REPLICATOR capabilities MAY be implemented as a system level option as opposed to as a per-MAC-VRF option.

b) An AR-REPLICATOR MUST advertise a Replicator-AR route and MAY advertise a Regular-IR route. The AR-REPLICATOR MUST NOT generate a Regular-IR route if it does not have local attachment circuits (AC).

c) The Replicator-AR and Regular-IR routes will be generated according to section 3. The AR-IP and IR-IP used by the Replicator-AR will be different routable IP addresses.

d) When a node defined as AR-REPLICATOR receives a packet on an overlay tunnel, it will do a tunnel destination IP lookup and apply the following procedures:

   - If the destination IP is the AR-REPLICATOR IR-IP Address the node will process the packet normally as in [RFC7432].

   - If the destination IP is the AR-REPLICATOR AR-IP Address the node MUST replicate the packet to local ACs and overlay tunnels (excluding the overlay tunnel to the source of the packet). When replicating to remote AR-REPLICATORs the tunnel destination IP will be an IR-IP. That will be an indication for the remote AR-REPLICATOR that it MUST NOT replicate to overlay tunnels. The tunnel source IP will be the AR-IP of the AR-REPLICATOR.

4.2. Non-selective AR-LEAF procedures

AR-LEAF is defined as an NVE/PE that - given its poor replication performance - sends all the BM traffic to an AR-REPLICATOR that can replicate the traffic further on its behalf. It MAY signal its AR-LEAF capability in the control plane and understands where the other roles are located (AR-REPLICATOR and RNVEs). A given service can have zero, one or more AR-LEAF nodes. Figure 1 shows NVE1 and NVE3 (both residing in hypervisors) acting as AR-LEAF. The following considerations apply to the AR-LEAF role:

a) The AR-LEAF role SHOULD be an administrative choice in any NVE/PE that is part of an AR-enabled EVI. This administrative option to enable AR-LEAF capabilities MAY be implemented as a system level option as opposed to as per-MAC-VRF option.

b) In this non-selective AR solution, the AR-LEAF MUST advertise a single Regular-IR inclusive multicast route as in [RFC7432]. The

AR-LEAF SHOULD set the AR Type field to AR-LEAF. Note that although this flag does not make any difference for the egress nodes when creating an EVPN destination to the the AR-LEAF, it is RECOMMENDED the use of this flag for an easy operation and troubleshooting of the EVI.

c) In a service where there are no AR-REPLICATORs, the AR-LEAF MUST use regular ingress replication. This will happen when a new update from the last former AR-REPLICATOR is received and contains a non-REPLICATOR AR type, or when the AR-LEAF detects that the last AR-REPLICATOR is down (next-hop tracking in the IGP or any other detection mechanism). Ingress replication MUST use the forwarding information given by the remote Regular-IR Inclusive Multicast Routes as described in [RFC7432].

d) In a service where there is one or more AR-REPLICATORs (based on the received Replicator-AR routes for the EVI), the AR-LEAF can locally select which AR-REPLICATOR it sends the BM traffic to:

- o A single AR-REPLICATOR MAY be selected for all the BM packets received on the AR-LEAF attachment circuits (ACs) for a given EVI. This selection is a local decision and it does not have to match other AR-LEAF's selection within the same EVI.
- o An AR-LEAF MAY select more than one AR-REPLICATOR and do either per-flow or per-EVI load balancing.
- o In case of a failure on the selected AR-REPLICATOR, another AR-REPLICATOR will be selected.
- o When an AR-REPLICATOR is selected, the AR-LEAF MUST send all the BM packets to that AR-REPLICATOR using the forwarding information given by the Replicator-AR route for the chosen AR-REPLICATOR, with tunnel type = 0x0A (AR tunnel). The underlay destination IP address MUST be the AR-IP advertised by the AR-REPLICATOR in the Replicator-AR route.
- o AR-LEAF nodes SHALL send service-level BM control plane packets following regular IR procedures. An example would be IGMP, MLD or PIM multicast packets. The AR-REPLICATORs MUST not replicate these control plane packets to other overlay tunnels since they will use the regular IR-IP Address.

e) The use of an AR-REPLICATOR-activation-timer (in seconds) on the AR-LEAF nodes is RECOMMENDED. Upon receiving a new Replicator-AR route where the AR-REPLICATOR is selected, the AR-LEAF will run a timer before programming the new AR-REPLICATOR. This will give the AR-REPLICATOR some time to program the AR-LEAF nodes before the

AR-LEAF sends BM traffic.

4.3. RNVE procedures

RNVE (Regular Network Virtualization Edge node) is defined as an NVE/PE without AR-REPLICATOR or AR-LEAF capabilities that does IR as described in [RFC7432]. The RNVE does not signal any AR role and is unaware of the AR-REPLICATOR/LEAF roles in the EVI. The RNVE will ignore the Flags in the Regular-IR routes and will ignore the Replicator-AR routes (due to an unknown tunnel type in the PTA) and the Leaf-AD routes (due to the IP-address-specific route-target).

This role provides EVPN with the backwards compatibility required in optimized-IR EVIs. Figure 1 shows NVE2 as RNVE.

4.4. Forwarding behavior in non-selective AR EVIs

In AR EVIs, BM (Broadcast and Multicast) traffic between two NVEs may follow a different path than unicast traffic. This solution proposes the replication of BM through the AR-REPLICATOR node, whereas unknown/known unicast will be delivered directly from the source node to the destination node without being replicated by any intermediate node. Unknown unicast SHALL follow the same path as known unicast traffic in order to avoid packet reordering for unicast applications and simplify the control and data plane procedures. Section 4.4.1. describes the expected forwarding behavior for BM traffic in nodes acting as AR-REPLICATOR, AR-LEAF and RNVE. Section 4.4.2. describes the forwarding behavior for unknown unicast traffic.

Note that known unicast forwarding is not impacted by this solution.

4.4.1. Broadcast and Multicast forwarding behavior

The expected behavior per role is described in this section.

4.4.1.1. Non-selective AR-REPLICATOR BM forwarding

The AR-REPLICATORs will build a flooding list composed of ACs and overlay tunnels to remote nodes in the EVI. Some of those overlay tunnels MAY be flagged as non-BM receivers based on the BM flag received from the remote nodes in the EVI.

- When an AR-REPLICATOR receives a BM packet on an AC, it will forward the BM packet to its flooding list (including local ACs and remote NVE/PEs), skipping the non-BM overlay tunnels.

- When an AR-REPLICATOR receives a BM packet on an overlay tunnel, it

will check the destination IP of the underlay IP header and:

- If the destination IP matches its AR-IP, the AR-REPLICATOR will forward the BM packet to its flooding list (ACs and overlay tunnels) excluding the non-BM overlay tunnels. The AR-REPLICATOR will do source squelching to ensure the traffic is not sent back to the originating AR-LEAF. If the encapsulation is MPLSoGRE (or MPLSoUDP) and the EVI label is not the bottom of the stack, the AR-REPLICATOR MUST copy the rest of the labels and forward them to the egress overlay tunnels.

- If the destination IP matches its IR-IP, the AR-REPLICATOR will skip all the overlay tunnels from the flooding list, i.e. it will only replicate to local ACs. This is the regular IR behavior described in [RFC7432].

4.4.1.2. Non-selective AR-LEAF BM forwarding

The AR-LEAF nodes will build two flood-lists:

1) Flood-list #1 - composed of ACs and an AR-REPLICATOR-set of overlay tunnels. The AR-REPLICATOR-set is defined as one or more overlay tunnels to the AR-IP Addresses of the remote AR-REPLICATOR(s) in the EVI. The selection of more than one AR-REPLICATOR is described in section 4.2. and it is a local AR-LEAF decision.

2) Flood-list #2 - composed of ACs and overlay tunnels to the remote IR-IP Addresses.

When an AR-LEAF receives a BM packet on an AC, it will check the AR-REPLICATOR-set:

- If the AR-REPLICATOR-set is empty, the AR-LEAF will send the packet to flood-list #2.

- If the AR-REPLICATOR-set is NOT empty, the AR-LEAF will send the packet to flood-list #1, where only one of the overlay tunnels of the AR-REPLICATOR-set is used.

When an AR-LEAF receives a BM packet on an overlay tunnel, will forward the BM packet to its local ACs and never to an overlay tunnel. This is the regular IR behavior described in [RFC7432].

4.4.1.3. RNVE BM forwarding

The RNVE is completely unaware of the AR-REPLICATORs, AR-LEAF nodes

and BM/U flags (that information is ignored). Its forwarding behavior is the regular IR behavior described in [RFC7432]. Any regular non-AR node is fully compatible with the RNVE role described in this document.

4.4.2. Unknown unicast forwarding behavior

The expected behavior is described in this section.

4.4.2.1. Non-selective AR-REPLICATOR/LEAF Unknown unicast forwarding

While the forwarding behavior in AR-REPLICATORs and AR-LEAF nodes is different for BM traffic, as far as Unknown unicast traffic forwarding is concerned, AR-LEAF nodes behave exactly in the same way as AR-REPLICATORs do.

The AR-REPLICATOR/LEAF nodes will build a flood-list composed of ACs and overlay tunnels to the IR-IP Addresses of the remote nodes in the EVI. Some of those overlay tunnels MAY be flagged as non-U (Unknown unicast) receivers based on the U flag received from the remote nodes in the EVI.

- When an AR-REPLICATOR/LEAF receives an unknown packet on an AC, it will forward the unknown packet to its flood-list, skipping the non-U overlay tunnels.

- When an AR-REPLICATOR/LEAF receives an unknown packet on an overlay tunnel will forward the unknown packet to its local ACs and never to an overlay tunnel. This is the regular IR behavior described in [RFC7432].

4.4.2.2. RNVE Unknown unicast forwarding

As described for BM traffic, the RNVE is completely unaware of the REPLICATORs, LEAF nodes and BM/U flags (that information is ignored). Its forwarding behavior is the regular IR behavior described in [RFC7432], also for Unknown unicast traffic. Any regular non-AR node is fully compatible with the RNVE role described in this document.

5. Selective Assisted-Replication (AR) Solution Description

Figure 1 is also used to describe the selective AR solution, however in this section we consider NVE2 as one more AR-LEAF for EVI-1. The solution is called "selective" because a given AR-REPLICATOR MUST replicate the BM traffic to only the AR-LEAF that requested the replication (as opposed to all the AR-LEAF nodes) and MAY replicate the BM traffic to the RNVEs. The same AR roles defined in section 4

are used here, however the procedures are slightly different.

The following sub-sections describe the differences in the procedures of AR-REPLICATOR/LEAFs compared to the non-selective AR solution. There is no change on the RNVEs.

5.1. Selective AR-REPLICATOR procedures

In our example in figure 1, PE1 and PE2 are defined as Selective AR-REPLICATORs. The following considerations apply to the Selective AR-REPLICATOR role:

a) The Selective AR-REPLICATOR capability SHOULD be an administrative choice in any NVE/PE that is part of an AR-enabled EVI, as the AR role itself. This administrative option MAY be implemented as a system level option as opposed to as a per-MAC-VRF option.

b) Each AR-REPLICATOR will build a list of AR-REPLICATOR, AR-LEAF and RNVE nodes (AR-LEAF nodes that sent only a regular-IR route are accounted as RNVEs by the AR-REPLICATOR). In spite of the ’Selective’ administrative option, an AR-REPLICATOR MUST NOT behave as a Selective AR-REPLICATOR if at least one of the AR-REPLICATORs has the L flag NOT set. If at least one AR-REPLICATOR sends a Replicator-AR route with L=0 (in the EVI context), the rest of the AR-REPLICATORs will fall back to non-selective AR mode.

b) The Selective AR-REPLICATOR MUST follow the procedures described in section 4.1, except for the following differences:

   - The Replicator-AR route MUST include L=1 (Leaf Information Required) in the Replicator-AR route. This flag is used by the AR-REPLICATORs to advertise their ’selective’ AR-REPLICATOR capabilities. In addition, the AR-REPLICATOR auto-configures its IP-address-specific import route-target as described in section 3.

   - The AR-REPLICATOR will build a ’selective’ AR-LEAF-set with the list of nodes that requested replication to its own AR-IP. For instance, assuming NVE1 and NVE2 advertise a Leaf-AD route with PE1’s IP-address-specific route-target and NVE3 advertises a Leaf-AD route with PE2’s IP-address-specific route-target, PE1 MUST only add NVE1/NVE2 to its selective AR-LEAF-set for EVI-1, and exclude NVE3.

   - When a node defined and operating as Selective AR-REPLICATOR receives a packet on an overlay tunnel, it will do a tunnel

destination IP lookup and if the destination IP is the AR-REPLICATOR AR-IP Address, the node MUST replicate the packet to:

+ local ACs
+ overlay tunnels in the Selective AR-LEAF-set (excluding the overlay tunnel to the source AR-LEAF).
+ overlay tunnels to the RNVEs if the tunnel source IP is the IR-IP of an AR-LEAF (in any other case, the AR-REPLICATOR MUST NOT replicate the BM traffic to remote RNVEs). In other words, the first-hop selective AR-REPLICATOR will replicate to all the RNVEs.
+ overlay tunnels to the remote Selective AR-REPLICATORs if the tunnel source IP is an IR-IP of its own AR-LEAF-set (in any other case, the AR-REPLICATOR MUST NOT replicate the BM traffic to remote AR-REPLICATORs), where the tunnel destination IP is the AR-IP of the remote Selective AR-REPLICATOR. The tunnel destination IP AR-IP will be an indication for the remote Selective AR-REPLICATOR that the packet needs further replication to its AR-LEAFs.

5.2. Selective AR-LEAF procedures

A Selective AR-LEAF chooses a single Selective AR-REPLICATOR per EVI and:

o Sends all the EVI BM traffic to that AR-REPLICATOR and
o Expects to receive the BM traffic for a given EVI from the same AR-REPLICATOR.

In the example of Figure 1, we consider NVE1/NVE2/NVE3 as Selective AR-LEAFs. NVE1 selects PE1 as its Selective AR-REPLICATOR. If that is so, NVE1 will send all its BM traffic for EVI-1 to PE1. If other AR-LEAF/REPLICATORs send BM traffic, NVE1 will receive that traffic from PE1. These are the differences in the behavior of a Selective AR-LEAF compared to a non-selective AR-LEAF:

a) The AR-LEAF role selective capability SHOULD be an administrative choice in any NVE/PE that is part of an AR-enabled EVI. This administrative option to enable AR-LEAF capabilities MAY be implemented as a system level option as opposed to as per-MAC-VRF option.

b) The AR-LEAF MAY advertise a Regular-IR route if there are RNVEs in the EVI. The Selective AR-LEAF MUST advertise a Leaf-AD route after receiving a Replicator-AR route with L=1. It is recommended that the Selective AR-LEAF waits for a timer t before sending the

Leaf-AD route, so that the AR-LEAF receives all the Replicator-AR routes for the EVI.

c) In a service where there is more than one Selective AR-REPLICATORs the Selective AR-LEAF MUST locally select a single Selective AR-REPLICATOR for the EVI. Once selected:

- The Selective AR-LEAF will send a Leaf-AD route including the Route-key and IP-address-specific route-target of the selected AR-REPLICATOR.
- The Selective AR-LEAF will send all the BM packets received on the attachment circuits (ACs) for a given EVI to that AR-REPLICATOR.
- In case of a failure on the selected AR-REPLICATOR, another AR-REPLICATOR will be selected and a new Leaf-AD update will be issued for the new AR-REPLICATOR. This new route will update the selective list in the new Selective AR-REPLICATOR. In case of failure on the active Selective AR-REPLICATOR, it is recommended for the Selective AR-LEAF to revert to IR behavior for a timer t to speed up the convergence. When the timer expires, the Selective AR-LEAF will resume its AR mode with the new Selective AR-REPLICATOR.

All the AR-LEAFs in an EVI are expected to be configured as either selective or non-selective. A mix of selective and non-selective AR-LEAFs SHOULD NOT coexist in the same EVI. In case there is a non-selective AR-LEAF, its BM traffic sent to a selective AR-REPLICATOR will not be replicated to other AR-LEAFs that are not in its Selective AR-LEAF-set.

5.3. Forwarding behavior in selective AR EVIs

This section describes the differences of the selective AR forwarding mode compared to the non-selective mode. Compared to section 4.4, there are no changes for the forwarding behavior in RNVEs or for unknown unicast traffic.

5.3.1. Selective AR-REPLICATOR BM forwarding

The Selective AR-REPLICATORs will build two flood-lists:

1) Flood-list #1 - composed of ACs and overlay tunnels to the remote nodes in the EVI, always using the IR-IPs in the tunnel destination IP addresses. Some of those overlay tunnels MAY be flagged as non-BM receivers based on the BM flag received from the remote nodes in the EVI.

2) Flood-list #2 - composed of ACs, a Selective AR-LEAF-set and a Selective AR-REPLICATOR-set, where:

   o The Selective AR-LEAF-set is composed of the overlay tunnels to the AR-LEAFs that advertise a Leaf-AD route for the local AR-REPLICATOR. This set is updated with every Leaf-AD route received/withdrawn from a new AR-LEAF.

   o The Selective AR-REPLICATOR-set is composed of the overlay tunnels to all the AR-REPLICATORs that send a Replicator-AR route with L=1. The AR-IP addresses are used as tunnel destination IP.

When a Selective AR-REPLICATOR receives a BM packet on an AC, it will forward the BM packet to its flood-list #1, skipping the non-BM overlay tunnels.

When a Selective AR-REPLICATOR receives a BM packet on an overlay tunnel, it will check the destination and source IPs of the underlay IP header and:

- If the destination IP matches its AR-IP and the source IP matches an IP of its own Selective AR-LEAF-set, the AR-REPLICATOR will forward the BM packet to its flood-list #2, as long as the list of AR-REPLICATORs for the EVI matches the Selective AR-REPLICATOR-set. If the Selective AR-REPLICATOR-set does not match the list of AR-REPLICATORs, the node reverts back to non-selective mode and flood-list #1 is used.

- If the destination IP matches its AR-IP and the source IP does not match any IP of its Selective AR-LEAF-set, the AR-REPLICATOR will forward the BM packet to flood-list #2 but skipping the AR-REPLICATOR-set.

- If the destination IP matches its IR-IP, the AR-REPLICATOR will use flood-list #1 but MUST skip all the overlay tunnels from the flooding list, i.e. it will only replicate to local ACs. This is the regular-IR behavior described in [RFC7432].

In any case, non-BM overlay tunnels are excluded from flood-lists and, also, source squelching is always done in order to ensure the traffic is not sent back to the originating source. If the encapsulation is MPLSoGRE (or MPLSoUDP) and the EVI label is not the bottom of the stack, the AR-REPLICATOR MUST copy the rest of the labels when forwarding them to the egress overlay tunnels.

5.3.2. Selective AR-LEAF BM forwarding

The Selective AR-LEAF nodes will build two flood-lists:

1) Flood-list #1 - composed of ACs and the overlay tunnel to the selected AR-REPLICATOR (using the AR-IP as the tunnel destination IP).

2) Flood-list #2 - composed of ACs and overlay tunnels to the remote IR-IP Addresses.

When an AR-LEAF receives a BM packet on an AC, it will check if there is any selected AR-REPLICATOR. If there is, flood-list #1 will be used. Otherwise, flood-list #2 will.

When an AR-LEAF receives a BM packet on an overlay tunnel, will forward the BM packet to its local ACs and never to an overlay tunnel. This is the regular IR behavior described in [RFC7432].

6. Pruned-Flood-Lists (PFL)

In addition to AR, the second optimization supported by this solution is the ability for the all the EVI nodes to signal Pruned-Flood-Lists (PFL). As described in section 3, an EVPN node can signal a given value for the BM and U PFL flags in the IR Inclusive Multicast Routes, where:

+ BM= Broadcast and Multicast (BM) flag. BM=1 means "prune-me" from the BM flood-list. BM=0 means regular behavior.

+ U= Unknown flag. U=1 means "prune-me" from the Unknown flood-list. U=0 means regular behavior.

The ability to signal these PFL flags is an administrative choice. Upon receiving a non-zero PFL flag, a node MAY decide to honor the PFL flag and remove the sender from the corresponding flood-list. A given EVI node receiving BUM traffic on an overlay tunnel MUST replicate the traffic normally, regardless of the signaled PFL flags.

This optimization MAY be used along with the AR solution.

6.1. A PFL example

In order to illustrate the use of the solution described in this document, we will assume that EVI-1 in figure 1 is optimized-IR enabled and:

o PE1 and PE2 are administratively configured as AR-REPLICATORs, due

to their high-performance replication capabilities. PE1 and PE2 will send a Replicator-AR route with BM/U flags = 00.

- o NVE1 and NVE3 are administratively configured as AR-LEAF nodes, due to their low-performance software-based replication capabilities. They will advertise a Regular-IR route with type AR-LEAF. Assuming both NVEs advertise all the attached VMs in EVPN as soon as they come up and don’t have any VMs interested in multicast applications, they will be configured to signal BM/U flags = 11 for EVI-1.

- o NVE2 is optimized-IR unaware; therefore it takes on the RNVE role in EVI-1.

Based on the above assumptions the following forwarding behavior will take place:

(1) Any BM packets sent from VM11 will be sent to VM12 and PE1. PE1 will forward further the BM packets to TS1, WAN link, PE2 and NVE2, but not to NVE3. PE2 and NVE2 will replicate the BM packets to their local ACs but we will avoid NVE3 having to replicate unnecessarily those BM packets to VM31 and VM32.

(2) Any BM packets received on PE2 from the WAN will be sent to PE1 and NVE2, but not to NVE1 and NVE3, sparing the two hypervisors from replicating unnecessarily to their local VMs. PE1 and NVE2 will replicate to their local ACs only.

(3) Any Unknown unicast packet sent from VM31 will be forwarded by NVE3 to NVE2, PE1 and PE2 but not NVE1. The solution avoids the unnecessary replication to NVE1, since the destination of the unknown traffic cannot be at NVE1.

(4) Any Unknown unicast packet sent from TS1 will be forwarded by PE1 to the WAN link, PE2 and NVE2 but not to NVE1 and NVE3, since the target of the unknown traffic cannot be at those NVEs.

7. AR Procedures for single-IP AR-REPLICATORS

The procedures explained in sections 4 (Non-selective AR) and 5 (Selective AR) assume that the AR-REPLICATOR can use two local routable IP addresses to terminate and originate NVO tunnels, i.e. IR-IP and AR-IP addresses. This is usually the case for PE-based AR-REPLICATOR nodes.

In some cases, the AR-REPLICATOR node does not support more than one IP address to terminate and originate NVO tunnels, i.e. the IR-IP and AR-IP are the same IP addresses. This may be the case in some

software-based or low-end AR-REPLICATOR nodes. If this is the case, the procedures in sections 4 and 5 must be modified in the following way:

- The Replicator-AR routes generated by the AR-REPLICATOR use an AR-IP that will match its IR-IP. In order to differentiate the data plane packets that need to use IR from the packets that must use AR forwarding mode, the Replicator-AR route must advertise a different VNI/VSID than the one used by the Regular-IR route. For instance, the AR-REPLICATOR will advertise AR-VNI along with the Replicator-AR route and IR-VNI along with the Regular-IR route. Since both routes have the same key, different RDs are needed for both routes.

- An AR-REPLICATOR will perform IR or AR forwarding mode for the incoming Overlay packets based on an ingress VNI lookup, as opposed to the tunnel IP DA lookup described in sections 4 and 5. Note that, when replicating to remote AR-REPLICATOR nodes, the use of the IR-VNI or AR-VNI advertised by the egress node will determine the IR or AR forwarding mode at the subsequent AR-REPLICATOR.

The rest of the procedures will follow what is described in sections 4 and 5.

8. AR Procedures and EVPN Multi-homing Split-Horizon

If VXLAN or NVGRE are used, and if the Split-horizon is based on the tunnel IP SA and "Local-Bias" as described in [EVPN-OVERLAY], the Split-horizon check will not work if there is an Ethernet-Segment shared between two AR-LEAF nodes, and the AR-REPLICATOR changes the tunnel IP SA of the packets with its own AR-IP.

In order to be compatible with the IP SA split-horizon check, the AR-REPLICATOR MAY keep the original received tunnel IP SA when replicating packets to a remote AR-LEAF or AR-REPLICATOR. This will allow DF (Designated Forwarder) AR-LEAF nodes to apply Split-horizon check procedures for BM packets, before sending them to the local Ethernet-Segment.

When EVPN is used for MPLS over GRE (or UDP), the ESI-label based split-horizon procedure as in [RFC7432] will not work for multi-homed Ethernet-Segments defined on AR-LEAF nodes. "Local-Bias" is recommended in this case, as in the case of VXLAN or NVGRE explained above. The "Local-Bias" and tunnel IP SA preservation mechanisms provide the required split-horizon behavior in non-selective or selective AR.

Note that if the AR-REPLICATOR implementation keeps the received

tunnel IP SA, the use of uRPF in the IP fabric based on the tunnel IP SA MUST be disabled.

9. Out-of-band distribution of Broadcast/Multicast traffic

The use of out-of-band mechanisms to distribute BM traffic between AR-REPLICATORS MAY be used.

10. Benefits of the optimized-IR solution

A solution for the optimization of Ingress Replication in EVPN is described in this document (optimized-IR). The solution brings the following benefits:

- o Optimizes the multicast forwarding in low-performance NVEs, by relaying the replication to high-performance NVEs (AR-REPLICATORs) and while preserving the packet ordering for unicast applications.
- o Reduces the flooded traffic in NVO networks where some NVEs do not need broadcast/multicast and/or unknown unicast traffic.
- o It is fully compatible with existing EVPN implementations and EVPN functions for NVO overlay tunnels. Optimized-IR NVEs and regular NVEs can be even part of the same EVI.
- o It does not require any PIM-based tree in the NVO core of the network.

11. Conventions used in this document

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 RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

12. Security Considerations

This section will be added in future versions.

13. IANA Considerations

IANA has allocated the following Border Gateway Protocol (BGP) Parameters:

1) Allocation in the P-Multicast Service Interface Tunnel (PMSI Tunnel) Tunnel Types registry:

Value	Meaning	Reference
0x0A	Assisted-Replication Tunnel	[This document]

2) Allocations in the P-Multicast Service Interface (PMSI) Tunnel Attribute Flags registry:

Value	Name	Reference
3-4	Assisted-Replication Type (T)	[This document]
5	Broadcast and Multicast (BM)	[This document]
6	Unknown (U)	[This document]

14. Terminology

Regular-IR: Refers to Regular Ingress Replication, where the source NVE/PE sends a copy to each remote NVE/PE part of the EVI.

AR-IP: IP address owned by the AR-REPLICATOR and used to differentiate the ingress traffic that must follow the AR procedures.

IR-IP: IP address used for Ingress Replication as in [RFC7432].

AR-VNI: VNI advertised by the AR-REPLICATOR along with the Replicator-AR route. It is used to identify the ingress packets that must follow AR procedures ONLY in the Single-IP AR-REPLICATOR case.

IR-VNI: VNI advertised along with the RT-3 for IR.

AR forwarding mode: for an AR-LEAF, it means sending an AC BM packet to a single AR-REPLICATOR with tunnel destination IP AR-IP. For an AR-REPLICATOR, it means sending a BM packet to a selective number or all the overlay tunnels when the packet was previously received from an overlay tunnel.

IR forwarding mode: it refers to the Ingress Replication behavior explained in [RFC7432]. It means sending an AC BM packet copy

to each remote PE/NVE in the EVI and sending an overlay BM packet only to the ACs and not other overlay tunnels.

PTA: PMSI Tunnel Attribute

RT-3: EVPN Route Type 3, Inclusive Multicast Ethernet Tag route

RT-11: EVPN Route Type 11, Leaf Auto-Discovery (AD) route

15. References

15.1 Normative References

[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, <http://www.rfc-editor.org/info/rfc6514>.

[RFC7432]Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <http://www.rfc-editor.org/info/rfc7432>.

[RFC7902]Rosen, E. and Morin, T., "Registry and Extensions for P-Multicast Service Interface Tunnel Attribute Flags", June 2016, <http://www.rfc-editor.org/info/rfc7902>.

[EVPN-BUM] Zhang et al., "Updates on EVPN BUM Procedures", draft-ietf-bess-evpn-bum-procedure-updates-01.txt, work in progress, December 2016.

15.2 Informative References

[EVPN-OVERLAY] Sajassi-Drake et al., "A Network Virtualization Overlay Solution using EVPN", draft-ietf-bess-evpn-overlay-07.txt, work in progress, December 2016.

16. Acknowledgments

The authors would like to thank Neil Hart, David Motz, Kiran Nagaraj, Dai Truong, Thomas Morin, Jeffrey Zhang and Shankar Murthy for their valuable feedback and contributions.

17. Authors’ Addresses

Jorge Rabadan (Editor)
Nokia
777 E. Middlefield Road
Mountain View, CA 94043 USA
Email: jorge.rabadan@nokia.com

Senthil Sathappan
Nokia
Email: senthil.sathappan@nokia.com

Mukul Katiyar
Juniper Networks
Email: mkatiyar@juniper.net

Wim Henderickx
Nokia
Email: wim.henderickx@nokia.com

Ravi Shekhar
Juniper Networks
Email: rshekhar@juniper.net

Nischal Sheth
Juniper Networks
Email: nsheth@juniper.net

Wen Lin
Juniper Networks
Email: wlin@juniper.net

Ali Sajassi
Cisco
Email: sajassi@cisco.com

Aldrin Isaac
Juniper
Email: aisaac@juniper.net

Mudassir Tufail
Citibank
mudassir.tufail@citi.com

BESS Working Group

Internet Draft
Intended Status: Proposed Standard
Expires: September 14, 2017

P. Brissette
A.Sajassi
Cisco System
H. Shah
Ciena Corporation
Z. Li
Huawei Technologies
I. Chen
Jabil
K. Tiruveedhula
Juniper Networks
I. Hussain
Infinera Corporation
J. Rabadan
Nokia

March 13, 2017

Yang Data Model for EVPN
draft-ietf-bess-evpn-yang-02

Abstract

This document describes a YANG data model for Ethernet VPN services. The model is agnostic of the underlay. It apply to MPLS as well as to VxLAN encapsulation. The model is also agnostic of the services including E-LAN, E-LINE and E-TREE services. Any "add-on" features such as EVPN IRB, EVPN overlay, etc. are for future investigation. This document mainly focuses on EVPN and Ethernet-Segment instance framework.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

Copyright and License Notice

Convention

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].

Table of Contents

1. Introduction 4
2. Specification of Requirements 5
3. EVPN YANG Model 5
  3.1. Overview 5
  3.2 Ethernet-Segment Model 6
  3.3 EVPN Model 6
4. YANG Module 7
  4.1 Ethernet Segment Yang Module 7
  4.2 EVPN Yang Module 9
5. Security Considerations 11
6. IANA Considerations 11
7. Acknowledgments 11
8. References 12
  8.1. Normative References 12
  8.2. Informative References 12
Authors' Addresses 12

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is a network management protocol that defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML or JSON tree format, and is used as a data modeling language for the NETCONF.

This document introduces a YANG data model for Ethernet VPN services (EVPN) [RFC7432], Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN) [RFC7623] as well as other WG draft such as EVPN-VPWS, etc. The EVPN services runs over MPLS and VxLAN underlay.

The Yang data model in this document defines Ethernet VPN based services. The model will leverage the definitions used in other IETF Yang draft such as L2VPN Yang.

The goal is to propose a data object model consisting of building blocks that can be assembled in different order to realize different EVPN-based services. The definition work is undertaken initially by a smaller working group with members representing various vendors and service providers. The EVPN basic framework consist of two modules: EVPN and Ethernet-Segment. These models are completely orthogonal. They usually work in pair but user can definitely use one or the other for its own need.

The data model is defined for following constructs that are used for managing the services:

- Configuration
- Operational State
- Executables (Actions)
- Notifications

The document is organized to first define the data model for the configuration, operational state, actions and notifications of EVPN and Ethernet-Segment.

The EVPN data object model defined in this document uses the instance centric approach whereby EVPN service attributes are specified for a given EVPN instance.

The Ethernet-Segment data object model defined in this document refer to a specific interface. That interface can be a physical interface, a bundle interface or virtual interface. The latter includes

pseudowires. The purpose of creating a separate module is due to the fact that it can be used without having the need to have EVPN configured as layer 2 service. For example, an access node can be dual-homed to two service nodes servicing a VPLS core. The access connectivity can be represented by an Ethernet-Segment where EVPN BGP DF election is performed over both service nodes. The core remains VPLS where no EVPN instance is required.

2. Specification of Requirements

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].

3. EVPN YANG Model

3.1. Overview

Two top level module, Ethernet-Segment and EVPN, are defined. The Ethernet-Segment contains a list of interface to which any Ethernet-Segment attributes are configured/applied.

The EVPN module has 2 main containers: common and instance. The first one has common attributes to all VPNs where as the latter has attributes specific to an EVI. This document state the scope of the EVPN object models definition. The following documents are within the scope. This is not an exhaustive list but a representation of documents that are covered for this work:

- Requirements for EVPN: RFC 7209
- EVPN: RFC 7432
- PBB-EVPN: RFC 7623

The integration with L2VPN instance Yang model is being done as part of the L2VPN Yang model.
Following documents will be covered at that time:

- VPWS support in EVPN:
draft-ietf-bess-evpn-vpws
- E-TREE Support in EVPN & PBB-EVPN:
draft-ietf-bess-evpn-etree
- (PBB-)EVPN Seamless Integration with (PBB-)VPLS:
draft-ietf-bess-evpn-vpls-seamless-integ
- EVPN Virtual Ethernet Segment:
draft-sajassi-bess-evpn-virtual-eth-segment

The VxLAN aspect and the work related to Layer 3 is also for future definition. Following documents will be covered at that time:

o IP Prefix Advertisement in EVPN:
draft-ietf-bess-evpn-prefix-advertisement
o VXLAN DCI Using EVPN:
draft-boutros-l2vpn-vxlan-evpn
o A Network Virtualization Overlay Solution using EVPN:
draft-ietf-bess-evpn-overlay-
o Interconnect Solution for EVPN Overlay networks:
draft-ietf-bess-dci-evpn-overlay
o Integrated Routing and Bridging in EVPN:
draft-ietf-bess-evpn-inter-subnet-forwarding

3.2 Ethernet-Segment Model

The Ethernet-Segment data model has a list of ES where each refer to an interface. All attributes are optional due to auto-sensing default mode where all values are auto-derive from the network connectivity.

```
module: ietf-ethernet-segment
    +--rw ethernet-segments
    |  +--rw ethernet-segment* [name]
    |     +--rw name                              string
    |     +--rw (ac-or-pw)?
    |     |  +--:(ac)
    |     |  |  +--rw ac?                             string
    |     |  +--:(pw)
    |     |     +--rw pw?                             string
    |     +--rw ethernet-segment-identifier?   uint32
    |     +--rw (active-mode)
    |     |  +--:(single-active)
    |     |  |  +--rw single-active-mode?            empty
    |     |  +--:(all-active)
    |     |     +--rw all-active-mode?               empty
    |     +--rw pbb-parameters {ethernet-segment-pbb-params}?
    |     |  +--rw backbone-src-mac?   yang:mac-address
    |     +--rw bgp-parameters
    |     |  +--rw common
    |     |     +--rw rd-rt* [route-distinguisher]
    |     |                  {ethernet-segment-bgp-params}?
    |     |        +--rw route-distinguisher
    |     |               rt-types:route-distinguisher
    |     |        +--rw vpn-target* [route-target]
    |     |           +--rw route-target
    |     |                 rt-types:route-target
    |     |           +--rw route-target-type
    |     |                 rt-types:route-target-type
    |     +--rw df-election
    |     |  +--rw (df-election-method)?
    |     |  |  +--:(highest-random-weight)
```

```
|     |  |     +--rw hrw?                  boolean
|     |  +--rw election-wait-time?   uint32
|     +--rw ead-evi-route?                boolean
+--ro ethernet-segments-state
   +--ro ethernet-segment-state* [name]
      +--ro name                           string
      +--ro service-type?                  string
      +--ro status?                        status-type
      +--ro (ac-or-pw)?
      |  +--:(ac)
      |  |  +--ro ac?                           string
      |  +--:(pw)
      |     +--ro pw?                           string
      +--ro interface-status?              status-type
      +--ro ethernet-segment-identifier?   uint32
      +--ro active-mode?                   string
      +--ro pbb-parameters {ethernet-segment-pbb-params}?
      |  +--ro backbone-src-mac?   yang:mac-address
      +--ro bgp-parameters
      |  +--ro common
      |     +--ro rd-rt* [route-distinguisher]
      |                  {ethernet-segment-bgp-params}?
      |        +--ro route-distinguisher
      |              rt-types:route-distinguisher
      |        +--ro vpn-target* [route-target]
      |           +--ro route-target
      |                 rt-types:route-target
      |           +--ro route-target-type
      |                 rt-types:route-target-type
      +--ro df-election
      |  +--ro hrw-enabled?          boolean
      |  +--ro election-wait-time?   uint32
      +--ro ead-evi-route-enabled?         boolean
      +--ro esi-label?                     string
      +--ro member*
      |  +--ro ip-address?   inet:ip-address
      +--ro df*
         +--ro service-identifier?   uint32
         +--ro vlan?                 uint32
         +--ro ip-address?           inet:ip-address
```

3.3 EVPN Model

The evpn-instances container contains a list of evpn-instance. Each entry of the evpn-instance represents a different Ethernet VPN and it is represented by a EVI. Again, mainly all attributes are optional for the same reason as for the Ethernet-Segment module.

```
   module: ietf-evpn
       +--rw evpn
       |  +--rw common
       |  |  +--rw (replication-type)?
       |  |     +--:(ingress-replication)
       |  |     |  +--rw ingress-replication?   boolean
       |  |     +--:(p2mp-replication)
       |  |        +--rw p2mp-replication?      boolean
       |  +--rw evpn-instances
       |     +--rw evpn-instance* [name]
       |        +--rw name                                  string
       |        +--rw evi?                                  uint32
       |        +--rw pbb-parameters {evpn-pbb-params}?
       |        |  +--rw source-bmac?   yang:hex-string
       |        +--rw bgp-parameters
       |        |  +--rw common
       |        |     +--rw rd-rt* [route-distinguisher]
       |        |                  {evpn-bgp-params}?
       |        |        +--rw route-distinguisher
       |        |               rt-types:route-distinguisher
       |        |        +--rw vpn-target* [route-target]
       |        |           +--rw route-target
       |        |                 rt-types:route-target
       |        |           +--rw route-target-type
       |        |                 rt-types:route-target-type
       |        +--rw arp-proxy?                            boolean
       |        +--rw arp-suppression?                      boolean
       |        +--rw nd-proxy?                             boolean
       |        +--rw nd-suppression?                       boolean
       |        +--rw underlay-multicast?                   boolean
       |        +--rw flood-unknown-unicast-supression?   boolean
       +--rw evpn-state
          +--ro evpn-instances-state
             +--ro evpn-instance*
                +--ro name? string
                +--ro evi? uint32
                +--ro pbb-parameters {evpn-pbb-params}?
                |  +--ro source-bmac?   yang:hex-string
                +--ro bgp-parameters
                |  +--ro common
                |     +--ro rd-rt* [route-distinguisher]
                |                  {evpn-bgp-params}?
                |        +--ro route-distinguisher
                |                 rt-types:route-distinguisher
                |        +--ro vpn-target* [route-target]
                |           +--ro route-target rt-types:route-target
                |           +--ro route-target-type
                |                 rt-types:route-target-type
```

```
               +--ro advertise-mac-suppression-enabled?         boolean
               +--ro arp-proxy-enabled?                         boolean
               +--ro arp-suppression-enabled?                   boolean
               +--ro nd-proxy-enabled?                          boolean
               +--ro nd-suppression-enabled?                    boolean
               +--ro underlay-multicast-enabled?                boolean
               +--ro flood-unknown-unicast-suppression-enabled? boolean
               +--ro routes
               |  +--ro ethernet-auto-discovery-route*
               |  |  +--ro rd-rt* [route-distinguisher]
               |  |  |  +--ro route-distinguisher
               |  |  |        rt-types:route-distinguisher
               |  |  |  +--ro vpn-target* [route-target]
               |  |  |     +--ro route-target    rt-types:route-target
               |  |  +--ro ethernet-segment-identifier?   uint32
               |  |  +--ro ethernet-tag?                  uint32
               |  |  +--ro path*
               |  |     +--ro next-hop?   inet:ip-address
               |  |     +--ro label?      rt-types:mpls-label
               |  |     +--ro detail
               |  |        +--ro attributes
               |  |        |  +--ro extended-community*   string
               |  |        +--ro bestpath?     empty
               |  +--ro mac-ip-advertisement-route*
               |  |  +--ro rd-rt* [route-distinguisher]
               |  |  |  +--ro route-distinguisher
               |  |  |        rt-types:route-distinguisher
               |  |  |  +--ro vpn-target* [route-target]
               |  |  |     +--ro route-target    rt-types:route-target
               |  |  +--ro ethernet-segment-identifier?   uint32
               |  |  +--ro ethernet-tag?                  uint32
               |  |  +--ro mac-address? yang:hex-string
               |  |  +--ro mac-address-length?            uint8
               |  |  +--ro ip-prefix?   inet:ip-prefix
               |  |  +--ro path*
               |  |     +--ro next-hop?   inet:ip-address
               |  |     +--ro label?      rt-types:mpls-label
               |  |     +--ro label2?     rt-types:mpls-label
               |  |     +--ro detail
               |  |        +--ro attributes
               |  |        |  +--ro extended-community*   string
               |  |        +--ro bestpath?     empty
               |  +--ro inclusive-multicast-ethernet-tag-route*
               |  |  +--ro rd-rt* [route-distinguisher]
               |  |  |  +--ro route-distinguisher
               |  |  |        rt-types:route-distinguisher
               |  |  |  +--ro vpn-target* [route-target]
               |  |  |     +--ro route-target    rt-types:route-target
```

```
|  |  +--ro ethernet-segment-identifier?   uint32
|  |  +--ro originator-ip-prefix? inet:ip-prefix
|  |  +--ro path*
|  |     +--ro next-hop?   inet:ip-address
|  |     +--ro label?      rt-types:mpls-label
|  |     +--ro detail
|  |        +--ro attributes
|  |        |  +--ro extended-community*   string
|  |        +--ro bestpath?     empty
|  +--ro ethernet-segment-route*
|  |  +--ro rd-rt* [route-distinguisher]
|  |  |  +--ro route-distinguisher
|  |  |        rt-types:route-distinguisher
|  |  |  +--ro vpn-target* [route-target]
|  |  |     +--ro route-target    rt-types:route-target
|  |  +--ro ethernet-segment-identifier?   uint32
|  |  +--ro originator-ip-prefix?     inet:ip-prefix
|  |  +--ro path*
|  |     +--ro next-hop?   inet:ip-address
|  |     +--ro detail
|  |        +--ro attributes
|  |        |  +--ro extended-community*   string
|  |        +--ro bestpath?     empty
|  +--ro ip-prefix-route*
|     +--ro rd-rt* [route-distinguisher]
|     |  +--ro route-distinguisher
|     |        rt-types:route-distinguisher
|     |  +--ro vpn-target* [route-target]
|     |     +--ro route-target    rt-types:route-target
|     +--ro ethernet-segment-identifier?   uint32
|     +--ro ip-prefix?  inet:ip-prefix
|     +--ro path*
|        +--ro next-hop?   inet:ip-address
|        +--ro label?      rt-types:mpls-label
|        +--ro detail
|           +--ro attributes
|           |  +--ro extended-community*   string
|           +--ro bestpath?     empty
+--ro statistics
   +--ro tx-count?   uint32
   +--ro rx-count?   uint32
   +--ro detail
      +--ro broadcast-tx-count?   uint32
      +--ro broadcast-rx-count?   uint32
      +--ro multicast-tx-count?   uint32
      +--ro multicast-rx-count?   uint32
      +--ro unicast-tx-count?     uint32
      +--ro unicast-rx-count?     uint32
```

```
augment /l2vpn:l2vpn/l2vpn:l2vpn-instances/l2vpn:l2vpn-instance:
  +--rw evpn-instance?   evpn-instance-ref
augment /l2vpn:l2vpn-state/
        l2vpn:l2vpn-instances-state/l2vpn:l2vpn-instance:
  +--ro evpn-instance?   string
```

4. YANG Module

The EVPN configuration container is logically divided into following high level config areas:

4.1 Ethernet Segment Yang Module

```
<CODE BEGINS> file "ietf-ethernet-segment@2017-03-13.yang"
module ietf-ethernet-segment {
  namespace "urn:ietf:params:xml:ns:yang:ietf-ethernet-segment";
  prefix "es";

  import ietf-yang-types {
    prefix "yang";
  }

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

  organization  "ietf";
  contact       "ietf";
  description   "ethernet segment";

  revision "2017-03-13" {
    description " - Updated to use BGP parameters from " +
                "   ietf-routing-types.yang instead of from " +
                "   ietf-evpn.yang " +
                "";
    reference   "";
  }

  revision "2016-07-08" {
    description "   - Added the configuration option to enable or " +
                "     disable per-EVI/EAD route " +
                "   - Added PBB parameter backbone-src-mac " +
                "   - Added operational state branch, initially " +
                "     to match the configuration branch" +
```

```
                "";
      reference   "";
    }

    revision "2016-06-23" {
      description "WG document adoption";
      reference   "";
    }

    revision "2015-10-15" {
      description "Initial revision";
      reference   "";
    }

    /* Features */

    feature ethernet-segment-bgp-params {
      description "Ethernet segment's BGP parameters";
    }

    feature ethernet-segment-pbb-params {
      description "Ethernet segment's PBB parameters";
    }

    /* Typedefs */

    typedef status-type {
      type enumeration {
        enum up {
          description "Status is up";
        }
        enum down {
          description "Status is down";
        }
      }
      description "status type";
    }

    /* EVPN Ethernet Segment YANG Model */

    container ethernet-segments {

      description "ethernet-segment";
      list ethernet-segment {
        key "name";
        leaf name {
          type string;
          description "Name of the ethernet segment";
```

```
        }
        choice ac-or-pw {
          description "ac-or-pw";
          case ac {
            leaf ac {
              type string;
              description "Eventual reference to standard " +
                          "attachment circuit definition";
            }
          }
          case pw {
            leaf pw {
              type string;
              description "Eventual reference to standard " +
                          "pseudowire definition";
            }
          }
        }
        leaf ethernet-segment-identifier {
          type uint32;
          description "Ethernet segment identifier (esi)";
        }
        choice active-mode {
          mandatory true;
          description "Choice of active mode";
          case single-active {
            leaf single-active-mode {
              type empty;
              description "single-active-mode";
            }
          }
          case all-active {
            leaf all-active-mode {
              type empty;
              description "all-active-mode";
            }
          }
        }
        container pbb-parameters {
          if-feature ethernet-segment-pbb-params;
          description "PBB configuration";
          leaf backbone-src-mac {
            type yang:mac-address;
            description "backbone-src-mac, only if this is a PBB";
          }
        }
        container bgp-parameters {
          description "BGP parameters";
```

```
          container common {
            description "BGP parameters common to all pseudowires";
            list rd-rt {
              if-feature ethernet-segment-bgp-params;
              key "route-distinguisher";
              leaf route-distinguisher {
                type rt-types:route-distinguisher;
                description "Route distinguisher";
              }
              uses rt-types:vpn-route-targets;
              description "A list of route distinguishers and " +
                          "corresponding VPN route targets";
            }
          }
        }
        container df-election {
          description "df-election";
          choice df-election-method {
            description "Choice of df election method";
            case highest-random-weight {
              leaf hrw {
                type boolean;
                description "Enable (TRUE) or disable (FALSE) " +
                            "highest random weight";
              }
            }
          }
          leaf election-wait-time {
            type uint32;
            description "election-wait-time";
          }
        }
        leaf ead-evi-route {
          type boolean;
          default false;
          description "Enable (true) or disable (false) ead-evi-route";
        }
        description "An ethernet segment";
      }
    }

    container ethernet-segments-state {
      config false;
      description "Ethernet segmet operational state";
      list ethernet-segment-state {
        key "name";
        leaf name {
          type string;
```

```
          description "Name of the ethernet segment";
        }
        leaf service-type {
          type string;
          description "service-type";
        }
        leaf status {
          type status-type;
          description "Ethernet segment status";
        }
        choice ac-or-pw {
          description "ac-or-pw";
          case ac {
            leaf ac {
              type string;
              description "Name of attachment circuit";
            }
          }
          case pw {
            leaf pw {
              type string;
              description "Name of pseudowire";
            }
          }
        }
        leaf interface-status {
          type status-type;
          description "interface status";
        }
        leaf ethernet-segment-identifier {
          type uint32;
          description "Ethernet segment identifier (esi)";
        }
        leaf active-mode {
          type string;
          description "Single-active-mode/all-active-mode";
        }
        container pbb-parameters {
          if-feature "ethernet-segment-pbb-params";
          description "PBB configuration";
          leaf backbone-src-mac {
            type yang:mac-address;
            description "backbone-src-mac, only if this is a PBB";
          }
        }
        container bgp-parameters {
          description "BGP parameters";
          container common {
```

```
            description "BGP parameters common to all pseudowires";
            list rd-rt {
              if-feature ethernet-segment-bgp-params;
              key "route-distinguisher";
              leaf route-distinguisher {
                type rt-types:route-distinguisher;
                description "Route distinguisher";
              }
              uses rt-types:vpn-route-targets;
              description "A list of route distinghishers and " +
                          "corresponding route targets";
            }
          }
        }
        container df-election {
          description "df-election";
          leaf hrw-enabled {
            type boolean;
            description "hrw-enabled is enabled (TRUE) " +
                        "or disabled (FALSE)";
          }
          leaf election-wait-time {
            type uint32;
            description "election-wait-time";
          }
        }
        leaf ead-evi-route-enabled {
          type boolean;
          description "ead-evi-route is enabled (TRUE) " +
                      "or disabled (FALSE)";
        }
        leaf esi-label {
          type string;
          description "esi-label";
        }
        list member {
          leaf ip-address {
            type inet:ip-address;
            description "ip-address";
          }
          description "member of the ethernet segment";
        }
        list df {
          leaf service-identifier {
            type uint32;
            description "service-identifier";
          }
          leaf vlan {
```

```
            type uint32;
            description "vlan";
          }
          leaf ip-address {
            type inet:ip-address;
            description "ip-address";
          }
          description "df of an evpn instance's vlan";
        }
        description "An ethernet segment";
      }
    }
  }
```

<CODE ENDS>

4.2 EVPN Yang Module

<CODE BEGINS> file "ietf-evpn@2017-03-13.yang"

```
module ietf-evpn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-evpn";
  prefix "evpn";

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-yang-types {
    prefix "yang";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

  organization  "ietf";
  contact       "ietf";
  description   "evpn";

  revision "2017-03-13" {
    description " - Added an augment to base L2VPN model to " +
                "   reference an EVPN instance " +
                " - Reused ietf-routing-types.yang " +
```

```
                "   vpn-route-targets grouping instead of " +
                "   defining it in this module " +
                "";
      reference   "";
    }

    revision "2016-07-08" {
      description " - Added operational state" +
                  " - Added a configuration knob to enable/disable " +
                  "   underlay-multicast " +
                  " - Added a configuration knob to enable/disable " +
                  "   flooding of unknonw unicast " +
                  " - Added several configuration knobs " +
                  "   to manage ARP and ND" +
                  "";
      reference   "";
    }

    revision "2016-06-23" {
      description "WG document adoption";
      reference   "";
    }

    revision "2015-10-15" {
      description "Initial revision";
      reference   "";
    }

    feature evpn-bgp-params {
      description "EVPN's BGP parameters";
    }

    feature evpn-pbb-params {
      description "EVPN's PBB parameters";
    }

    /* Typedefs */

    typedef evpn-instance-ref {
      type leafref {
        path "/evpn/evpn-instances/evpn-instance/name";
      }
      description "A leafref type to an EVPN instance";
    }

    /* Groupings */

    grouping route-rd-rt-grp {
```

```
      description "A grouping for a route's route distinguishers " +
                  "and route targets";
      list rd-rt {
        key "route-distinguisher";
        leaf route-distinguisher {
          type rt-types:route-distinguisher;
          description "Route distinguisher";
        }
        list vpn-target {
          key "route-target";
          leaf route-target {
            type rt-types:route-target;
            description "BGP route target";
          }
          description "A list of route targets";
        }
        description "A list of route distinguishers and " +
                    "corresponding VPN route targets";
      }
    }

    grouping next-hop-label-grp {
      description "next-hop-label-grp";
      leaf next-hop {
        type inet:ip-address;
        description "next-hop";
      }
      leaf label {
        type rt-types:mpls-label;
        description "label";
      }
    }

    grouping next-hop-label2-grp {
      description "next-hop-label2-grp";
      leaf label2 {
        type rt-types:mpls-label;
        description "label2";
      }
    }

    grouping path-detail-grp {
      description "path-detail-grp";
      container detail {
        config false;
        description "path details";
        container attributes {
          leaf-list extended-community {
```

```
            type string;
            description "extended-community";
          }
          description "attributes";
        }
        leaf bestpath {
          type empty;
          description "Indicate this path is the best path";
        }
      }
    }

    /* EVPN YANG Model */

    container evpn {
      description "evpn";
      container common {
        description "common epn attributes";
        choice replication-type {
          description "A choice of replication type";
          case ingress-replication {
            leaf ingress-replication {
              type boolean;
              description "ingress-replication";
            }
          }
          case p2mp-replication {
            leaf p2mp-replication {
              type boolean;
              description "p2mp-replication";
            }
          }
        }
      }
      container evpn-instances {
        description "evpn-instances";
        list evpn-instance {
          key "name";
          description "An EVPN instance";
          leaf name {
            type string;
            description "Name of EVPN instance";
          }
          leaf evi {
            type uint32;
            description "evi";
          }
          container pbb-parameters {
```

```
            if-feature "evpn-pbb-params";
            description "PBB parameters";
            leaf source-bmac {
              type yang:hex-string;
              description "source-bmac";
            }
          }
          container bgp-parameters {
            description "BGP parameters";
            container common {
              description "BGP parameters common to all pseudowires";
              list rd-rt {
                if-feature evpn-bgp-params;
                key "route-distinguisher";
                leaf route-distinguisher {
                  type rt-types:route-distinguisher;
                  description "Route distinguisher";
                }
                uses rt-types:vpn-route-targets;
                description "A list of route distinguishers and " +
                            "corresponding VPN route targets";
              }
            }
          }
          leaf arp-proxy {
            type boolean;
            default false;
            description "Enable (TRUE) or disable (FALSE) ARP proxy";
          }
          leaf arp-suppression {
            type boolean;
            default false;
            description "Enable (TRUE) or disable (FALSE) " +
                        "ARP suppression";
          }
          leaf nd-proxy {
            type boolean;
            default false;
            description "Enable (TRUE) or disable (FALSE) ND proxy";
          }
          leaf nd-suppression {
            type boolean;
            default false;
            description "Enable (TRUE) or disable (FALSE) " +
                        "ND suppression";
          }
          leaf underlay-multicast {
            type boolean;
```

```
            default false;
            description "Enable (TRUE) or disable (FALSE) " +
                        "underlay multicast";
          }
          leaf flood-unknown-unicast-supression {
            type boolean;
            default false;
            description "Enable (TRUE) or disable (FALSE) " +
                        "flood unknown unicast suppression";
          }
        }
      }
    }

    container evpn-state {
      description "EVPN operational state";
      container evpn-instances-state {
        config false;
        description "evpn-instances-state";
        list evpn-instance {
          description "The state of an EVPN instance";
          leaf name {
            type string;
            description "Name of EVPN instance";
          }
          leaf evi {
            type uint32;
            description "evi";
          }
          container pbb-parameters {
            if-feature "evpn-pbb-params";
            description "PBB parameters";
            leaf source-bmac {
              type yang:hex-string;
              description "source-bmac";
            }
          }
          container bgp-parameters {
            description "BGP parameters";
            container common {
              description "BGP parameters common to all pseudowires";
              list rd-rt {
                if-feature evpn-bgp-params;
                key "route-distinguisher";
                leaf route-distinguisher {
                  type rt-types:route-distinguisher;
                  description "Route distinguisher";
                }
```

```
                uses rt-types:vpn-route-targets;
                description "A list of route distinguishers and " +
                            "corresponding VPN route targets";
              }
            }
          }
          leaf advertise-mac-suppression-enabled {
            type boolean;
            description "advertise-mac-suppression " +
                        "is enabled (TRUE) " +
                        "or disabled (FALSE)";
          }
          leaf arp-proxy-enabled {
            type boolean;
            description "arp-proxy is enabled (TRUE) " +
                        "or disabled (FALSE)";
          }
          leaf arp-suppression-enabled {
            type boolean;
            description "arp-suppression is enabled (TRUE) " +
                        "or disabled (FALSE)";
          }
          leaf nd-proxy-enabled {
            type boolean;
            description "nd-proxy is enabled (TRUE) " +
                        "or disabled (FALSE)";
          }
          leaf nd-suppression-enabled {
            type boolean;
            description "nd-suppression is enabled (TRUE) " +
                        "or disabled (FALSE)";
          }
          leaf underlay-multicast-enabled {
            type boolean;
            description "underlay-multicast is enabled (TRUE) " +
                        "or disabled (FALSE)";
          }
          leaf flood-unknown-unicast-suppression-enabled {
            type boolean;
            description "flood-unknown-unicast-suppression is " +
                        "enabled (TRUE) or disabled (FALSE)";
          }
          container routes {
            description "routes";
            list ethernet-auto-discovery-route {
              uses route-rd-rt-grp;
              leaf ethernet-segment-identifier {
                type uint32;
```

```
                description "Ethernet segment identifier (esi)";
              }
              leaf ethernet-tag {
                type uint32;
                description "An ethernet tag (etag) indentifying a " +
                            "broadcast domain";
              }
              list path {
                uses next-hop-label-grp;
                uses path-detail-grp;
                description "path";
              }
              description "ethernet-auto-discovery-route";
            }
            list mac-ip-advertisement-route {
              uses route-rd-rt-grp;
              leaf ethernet-segment-identifier {
                type uint32;
                description "Ethernet segment identifier (esi)";
              }
              leaf ethernet-tag {
                type uint32;
                description "An ethernet tag (etag) indentifying a " +
                            "broadcast domain";
              }
              leaf mac-address {
                type yang:hex-string;
                description "Route mac address";
              }
              leaf mac-address-length {
                type uint8 {
                  range "0..48";
                }
                description "mac address length";
              }
              leaf ip-prefix {
                type inet:ip-prefix;
                description "ip-prefix";
              }
              list path {
                uses next-hop-label-grp;
                uses next-hop-label2-grp;
                uses path-detail-grp;
                description "path";
              }
              description "mac-ip-advertisement-route";
            }
            list inclusive-multicast-ethernet-tag-route {
```

```
            uses route-rd-rt-grp;
            leaf ethernet-segment-identifier {
              type uint32;
              description "Ethernet segment identifier (esi)";
            }
            leaf originator-ip-prefix {
              type inet:ip-prefix;
              description "originator-ip-prefix";
            }
            list path {
              uses next-hop-label-grp;
              uses path-detail-grp;
              description "path";
            }
            description "inclusive-multicast-ethernet-tag-route";
          }
          list ethernet-segment-route {
            uses route-rd-rt-grp;
            leaf ethernet-segment-identifier {
              type uint32;
              description "Ethernet segment identifier (esi)";
            }
            leaf originator-ip-prefix {
              type inet:ip-prefix;
              description "originator ip-prefix";
            }
            list path {
              leaf next-hop {
                type inet:ip-address;
                description "next-hop";
              }
              uses path-detail-grp;
              description "path";
            }
            description "ethernet-segment-route";
          }
          list ip-prefix-route {
            uses route-rd-rt-grp;
            leaf ethernet-segment-identifier {
              type uint32;
              description "Ethernet segment identifier (esi)";
            }
            leaf ip-prefix {
              type inet:ip-prefix;
              description "ip-prefix";
            }
            list path {
              uses next-hop-label-grp;
```

```
                uses path-detail-grp;
                description "path";
              }
              description "ip-prefix route";
            }
          }
          container statistics {
            description "Statistics";
            leaf tx-count {
              type uint32;
              description "transmission count";
            }
            leaf rx-count {
              type uint32;
              description "receive count";
            }
            container detail {
              description "Detailed statistics";
              leaf broadcast-tx-count {
                type uint32;
                description "broadcast transmission count";
              }
              leaf broadcast-rx-count {
                type uint32;
                description "broadcast receive count";
              }
              leaf multicast-tx-count {
                type uint32;
                description "multicast transmission count";
              }
              leaf multicast-rx-count {
                type uint32;
                description "multicast receive count";
              }
              leaf unicast-tx-count {
                type uint32;
                description "unicast transmission count";
              }
              leaf unicast-rx-count {
                type uint32;
                description "unicast receive count";
              }
            }
          }
        }
      }
    }
```

```
    /* augments */
    augment "/l2vpn:l2vpn/l2vpn:l2vpn-instances" +
            "/l2vpn:l2vpn-instance" {
      description "Augment for an L2VPN instance and EVPN association";
      leaf evpn-instance {
        type evpn-instance-ref;
        description "Reference to an EVPN instance";
      }
    }

    augment "/l2vpn:l2vpn-state" {
      description "Augment for an L2VPN instance's operational " +
                  "state of L2VPN and EVPN association";
      leaf evpn-instance {
        type string;
        description "Name of the associated EVPN instance";
      }
    }
  }
```

<CODE ENDS>

5. Security Considerations

The configuration, state, action and notification data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

The security concerns listed above are, however, no different than faced by other routing protocols. Hence, this draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg]

6. IANA Considerations

None.

7. Acknowledgments

The authors would like to acknowledge TBD for their useful comments.

8. References

8.1. Normative References

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

8.2. Informative References

[RFC6241] R.Enns et al., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011

[RFC6020] M. Bjorklund, "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.

[RFC6242] M. Wasserman, "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.

[RFC6536] A. Bierman et al., "Network Configuration Protocol (NETCONF) Access Control Model" RFC 6536, March 2012.

[RFC7432] Sajassi et al., "BGP MPLS-Based Ethernet VPN", RFC 7432, February 2015.

[RFC7623] Sajassi et al., "Provider Backbone Bridging Combined with Ethernet VPN (PBB-EVPN)", RFC 7623, September 2015

Authors' Addresses

Patrice Brissette
Cisco Systems, Inc.
EMail: pbrisset@cisco.com

Ali Sajassi
Cisco Systems, Inc.
EMail: sajassi@cisco.com

Himanshu Shah
Ciena Corporation
EMail: hshah@ciena.com

Zhenbin Li

Huawei Technologies
EMail: lizhenbin@huawei.com

Helen Chen
Jabil
EMail: Ing-Wher_Chen@jabil.com

Kishore Tiruveedhula
Juniper Networks
EMail: kishoret@juniper.net

Iftekar Hussain
Infinera Corporation
EMail: ihussain@infinera.com

Jorge Rabadan
Nokia
EMail: jorge.rabadan@nokia.com

BESS Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2018

H. Shah, Ed.
Ciena Corporation
P. Brissette, Ed.
Cisco Systems, Inc.
I. Chen, Ed.
Jabil
I. Hussain, Ed.
Infinera Corporation
B. Wen, Ed.
Comcast
K. Tiruveedhula, Ed.
Juniper Networks
June 30, 2017

YANG Data Model for MPLS-based L2VPN
draft-ietf-bess-l2vpn-yang-06.txt

Abstract

This document describes a YANG data model for Layer 2 VPN (L2VPN) services over MPLS networks. These services include point-to-point Virtual Private Wire Service (VPWS) and multipoint Virtual Private LAN service (VPLS) that uses LDP and BGP signaled Pseudowires. It is expected that this model will be used by the management tools run by the network operators in order to manage and monitor the network resources that they use to deliver L2VPN services.

This document also describes the YANG data model for the Pseudowires. The independent definition of the Pseudowires facilitates its use in Ethernet Segment and EVPN data models defined in separate document.

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 http://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 January 1, 2018.

Table of Contents

1. Introduction 3
2. Specification of Requirements 4
3. L2VPN YANG Model 4
3.1. Overview 4
3.2. Open issues and next steps 7
3.3. Pseudowire Common 8
3.3.1. Pseudowire 8
3.3.2. pw-templates 8
3.4. L2VPN Common 8
3.4.1. redundancy-group-templates 8
3.5. L2VPN instance 8
3.5.1. common attributes 8
3.5.2. PW list 8
3.5.3. List of endpoints 9
3.5.4. point-to-point or multipoint service 10
3.6. Operational State 10
3.7. Yang tree 10
4. YANG Module 13
5. Security Considerations 40
6. IANA Considerations 41
7. Acknowledgments 41
8. References 41
8.1. Normative References 41
8.2. Informative References 41
Appendix A. Example Configuration 44
Appendix B. Contributors 44
Authors' Addresses 45

1. Introduction

The Network Configuration Protocol (NETCONF) [RFC6241] is a network management protocol that defines mechanisms to manage network devices. YANG [RFC6020] is a modular language that represents data structures in an XML or JSON tree format, and is used as a data modeling language for the NETCONF.

This document defines a YANG data model for MPLS based Layer 2 VPN services (L2VPN) [RFC4664] and includes switching between the local attachment circuits. The L2VPN model covers point-to-point VPWS and Multipoint VPLS services. These services use signaling of Pseudowires across MPLS networks using LDP [RFC4447][RFC4762] or BGP[RFC4761].

Initially, the data model covers Ethernet based Layer 2 services. The Ethernet Attachment Circuits are not defined. Instead, they are leveraged from other standards organizations such as IEEE802.1 and Metro Ethernet Forum (MEF).

Other Layer 2 services, such as ATM, Frame Relay, TDM, etc are included in the scope but will be covered as the future work items.

The objective of the model is to define building blocks that can be easily assembled in different order to realize different services.

The data model uses following constructs for configuration and management:

- Configuration
- Operational State
- Executables (Actions)
- Notifications

The current document focuses on definition of configuration, state and notificationobjects.

The L2VPN data object model uses the instance centric approach. Within an L2VPN instance; a set of common parameters, a list of PWs and a list of endpoints are defined. A special constraint is added for the VPWS configuration such that only two endpoints are allowed in the list of endpoints.

The Pseudowire data object model is defined independent of the L2VPN data object model to allow its inclusion in the Ethernet Segment and EVPN data objects.

The L2VPN data object model augments Psuedowire data object for its definition.

The document also includes Notifications used by the L2VPN object model

2. Specification of Requirements

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].

3. L2VPN YANG Model

3.1. Overview

In this version of the document, for configuration, one single container, l2vpn, is defined. Within the l2vpn container, common parameters and a list of endpoints are defined. For the point-to-point VPWS configuration, endpoint list is used with the constraint that limits the number of endpoints to be two. For the multipoint service, endpoint list is used. Each endpoint contains the common definition that is either an attachment circuit, a pseudowire or a redundancy group. The YANG data model for l2vpn in this document is greatly simplified by by removing separate definition of endpoint-a and endpoint-z that was specific for VPWS service. The same endpoint list is used by both the VPLS and VPWS service with the exception that VPWS uses only two entries.

The l2vpn container also includes definition of common building blocks for redundancy-grp templates and pseudowire-templates.

The State objects have been consolidated with the configuration object as per the recommendations provided by the Guidelines for Yang Module Authors document.

The IETF working group has defined the VPWS and VPLS services that leverages the pseudowire technologies defined by the PWE3 working group. A large number of RFCs from these working groups cover this subject matter. Hence, it is prudent that this document state the scope of the MPLS L2VPN object model definitions.

The following documents are within the scope. This is not an exhaustive list but a representation of documents that are covered for this work:

- Requirements for Pseudo-wire Emulation Edge-to-Edge (PWE3) [RFC3916]
- Pseudo-wire Emulation Edge-to-Edge (PWE3) Architecture [RFC3985]
- IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3) [RFC4446]
- Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP) [RFC4447]
- Encapsulation Methods for Transport of Ethernet over MPLS Networks [RFC4448]
- Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN [RFC4385]
- Requirements for Multi-Segment Pseudowire Emulation Edge-to-Edge (PWE3) [RFC5254]
- An Architecture for Multi-Segment Pseudowire Emulation Edge-to-Edge [RFC5659]
- Segmented Pseudowire [RFC6073]
- Framework for Layer 2 Virtual Private Networks [RFC4664]
- Service Requirements for Layer 2 Provider-Provisioned Virtual Private Networks [RFC4665]
- Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling [RFC4761]
- Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling [RFC4762]
- Attachment Individual Identifier (AII) Types for Aggregation [RFC5003]
- Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs) [RFC6074]
- Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network [RFC6391]

- Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling [RFC6624]

- Extensions to the Virtual Private LAN Service (VPLS) Provider Edge (PE) Model for Provider Backbone Bridging [RFC7041]

- LDP Extensions for Optimized MAC Address Withdrawal in a Hierarchical Virtual Private LAN Service (H-VPLS) [RFC7361]

- Using the generic associated channel label for Pseudowire in the MPLS Transport Profile [RFC6423]

- Pseudowire status for static pseudowire [RFC6478]

The specifics of pseudowire over MPLS-TP LSPs is in scope. However, the initial effort addresses definitions of object models that are commonly deployed.

The IETF work in L2VPN and PWE3 working group relating to L2TP, OAM, multicast (e.g. p2mp, etree, etc) and access specific protocols such as G.8032, MSTP, etc is out-of-scope for this document.

The following is the high level view of the L2VPN data model.

```
 PW // Container
         PW specific attributes

         PW template definition


 template-ref Redundancy-Group // redundancy-group
                    template
                    attributes

 l2vpn-instances // containter

         common attributes


         BGP-parameters // container
                           common attributes
                           auto-discovery attributes
                           signaling attributes

         // list of PWs being used
         PW // container
                 template-ref PW
                 attribute-override

         PBB-parameters // container
                           pbb specific attributes

         VPWS-constraints // rule to limit number of endpoints to two

         // List of endpoints, where each member endpoint container is -
                 PW // reference
                 redundancy-grp // container
                         AC // eventual reference to standard AC
                         PW // reference
```

Figure 1

3.2. Open issues and next steps

Most of the open issues have been resolved in this document. There are some items for considerations, such as PW headend, VPLS IRB. These may or may not be convered in this document. If the working group intends these topics be addressed in a separate document, authors will proceed to finalize this document with comments received on the definitions included in the current document.

3.3. Pseudowire Common

3.3.1. Pseudowire

Pseudowire definitions is moved to a seperate container in order to allow Ethernet Segment and EVPN models can refer without having to pull down L2VPN container.

3.3.2. pw-templates

The pw-templates container contains a list of pw-template. Each pw-template defines a list of common pseudowire attributes such as PW MTU, control word support etc.

3.4. L2VPN Common

3.4.1. redundancy-group-templates

The redundancy-group-template contains a list of templates. Each template defines common attributes related to redundancy such as protection mode, reversion parameters, etc.

3.5. L2VPN instance

A list of L2VPN instance is defined where each entry represent a point to point or multipoint service. Within a service instance, a set of common attributes are defined, followed by a list of PWs and a list of endpoints.

3.5.1. common attributes

The common attributes apply to entire L2VPN instance. These attributes typically include attributes such as mac-aging-timer, BGP related parameters (if using BGP signaling), discovery-type, etc.

3.5.2. PW list

The PW list is the number of PWs that are being used for a given L2VPN instance. Each PW entry refers to PW template to inherit common attributes for the PW. The one or more attributes from the template can be overriden. It further extends definitions of more PW specific attributes such as use of control word, mac withdraw, what type of signaling (i.e. LDP or BGP), setting of the TTL, etc.

3.5.3. List of endpoints

The list of endpoints define the characteristics of the L2VPN service. In the case of VPWS, the list is limited to two entries while for VPLS, there could be many.

Each entry in the endpoint list, may hold AC, PW or redundancy-grp references. The core aspect of endpoint container is its flexible personality based on what user decides to include in it. It is future-proofed with possible extensions that can be included in the endpoint container such as Integrated Route Bridging (IRB), PW Headend, Virtual Switch Instance, etc.

The endpoint entry also defines the split-horizon attribute which defines the frame forwarding restrictions between the endpoints belonging to same split-horizon group. This construct permits multiple instances of split horizon groups with its own endpoint members. The frame forwarding restrictions does not apply between endpoints that belong to two different split horizon groups.

3.5.3.1. ac

Attachment Circuit (AC)resides within endpoint entry either as an independent entity or as a member of the redundancy group. AC is not defined in this document but references the definitions being specified by other working groups and standard bodies.

3.5.3.2. pw

The Pseudo-wire resides within endpoint entry either as an independent entity or as a member of the redundancy group. The PW refers to one of the entry in the list of PWs defined with the L2VPN instance.

3.5.3.3. redundancy-grp choice

The redundancy-grp is a generic redundancy construct which can hold primary and backup members of AC and PWs. This flexibility permits combinations of -

- primary and backup AC
- primary and backup PW
- primary AC and backup PW
- primary PW and backup AC

The redundancy group also defines attributes of the type of redundancy, such as protection mode, reroute mode, reversion related parameters, etc.

3.5.4. point-to-point or multipoint service

The point-to-point service as defined for VPWS is represented by a list of endpoints and is limited to two entries by the VPWS constrain rules

The multipoint service as defined for VPLS is represented by a list of endpoints.

The augmentation of ietf-l2vpn module is TBD. All IP addresses defined in this module are currently scoped under global VRF/table.

3.6. Operational State

The operational state of L2VPN attributes has been consolidated with the configuration as per recommendations from the guidelines for the YANG author document.

3.7. Yang tree

```
module: ietf-pseudowires
    +--rw pseudowires
       +--rw pseudowire* [name]
       |  +--rw name              string
       |  +--ro state?            pseudowire-status-type
       |  +--rw template?         pw-template-ref
       |  +--rw mtu?              uint16
       |  +--rw mac-withdraw?     boolean
       |  +--rw cw-negotiation?   cw-negotiation-type
       |  +--rw tunnel-policy?    string
       |  +--rw (pw-type)?
       |     +--:(configured-pw)
       |        +--rw configured-pw
       |           +--rw peer-ip?          inet:ip-address
       |           +--rw pw-id?            uint32
       |           +--rw icb?              boolean
       |           +--rw transmit-label?   rt-types:mpls-label
       |           +--rw receive-label?    rt-types:mpls-label
       +--rw pw-templates
          +--rw pw-template* [name]
             +--rw name              string
             +--rw mtu?              uint16
             +--rw cw-negotiation?   cw-negotiation-type
             +--rw tunnel-policy?    string
```

```
module: ietf-l2vpn
    +--rw l2vpn
       +--rw redundancy-group-templates
       |  +--rw redundancy-group-template* [name]
       |     +--rw name                string
       |     +--rw protection-mode?    enumeration
       |     +--rw reroute-mode?       enumeration
       |     +--rw dual-receive?       boolean
       |     +--rw revert?             boolean
       |     +--rw reroute-delay?      uint16
       |     +--rw revert-delay?       uint16
       +--rw instances
          +--rw instance* [name type]
             +--rw name                    string
             +--rw type                    identityref
             +--rw mtu?                    uint16
             +--rw mac-aging-timer?        uint32
             +--rw service-type?           l2vpn-service-type
             +--rw discovery-type?         l2vpn-discovery-type
             +--rw signaling-type          l2vpn-signaling-type
             +--rw bgp-auto-discovery
             |  +--rw route-distinguisher?   rt-types:route-distinguisher
             |  +--rw vpn-id?                string
             |  +--rw vpn-target* [route-target]
             |     +--rw route-target         rt-types:route-target
             |     +--rw route-target-type    rt-types:route-target-type
             +--rw bgp-signaling
             |  +--rw site-id?      uint16
             |  +--rw site-range?   uint16
             +--rw endpoint* [name]
             |  +--rw name                    string
             |  +--rw (ac-or-pw-or-redundancy-grp)?
             |  |  +--:(ac)
             |  |  |  +--rw ac* [name]
             |  |  |     +--rw name     string
             |  |  |     +--ro state?   operational-state-type
             |  |  +--:(pw)
             |  |  |  +--rw pw* [name]
             |  |  |     +--rw name     pw:pseudowire-ref
             |  |  |     +--ro state?   -> /pw:pseudowires/pseudowire[pw:name=cu
rrent()/../name]/state
             |  |  +--:(redundancy-grp)
             |  |     +--rw (primary)
             |  |     |  +--:(primary-ac)
             |  |     |  |  +--rw primary-ac
             |  |     |  |     +--rw name?    string
             |  |     |  |     +--ro state?   operational-state-type
             |  |     |  +--:(primary-pw)
             |  |     |     +--rw primary-pw* [name]
```

```
             |  |     |        +--rw name     pw:pseudowire-ref
             |  |     |        +--ro state?   -> /pw:pseudowires/pseudowire[pw:n
ame=current()/../name]/state
             |  |     +--rw (backup)?
             |  |     |  +--:(backup-ac)
             |  |     |  |  +--rw backup-ac
             |  |     |  |     +--rw name?    string
             |  |     |  |     +--ro state?   operational-state-type
             |  |     |  +--:(backup-pw)
             |  |     |     +--rw backup-pw* [name]
             |  |     |        +--rw name          pw:pseudowire-ref
             |  |     |        +--ro state?        -> /pw:pseudowires/pseudowire
[pw:name=current()/../name]/state
             |  |     |        +--rw precedence?   uint32
             |  |     +--rw template?              -> /l2vpn/redundancy-group-te
mplates/redundancy-group-template/name
             |  |     +--rw protection-mode?       enumeration
             |  |     +--rw reroute-mode?          enumeration
             |  |     +--rw dual-receive?          boolean
             |  |     +--rw revert?                boolean
             |  |     +--rw reroute-delay?         uint16
             |  |     +--rw revert-delay?          uint16
             |  +--rw split-horizon-group?   string
            +--rw vpws-constraints
            +--rw pbb-parameters
               +--rw (component-type)?
                  +--:(i-component)
                  |  +--rw i-sid?                   i-sid-type
                  |  +--rw backbone-src-mac?        yang:mac-address
                  +--:(b-component)
                     +--rw bind-b-component-name?   l2vpn-instance-name-ref
                     +--ro bind-b-component-type?   identityref
  augment /pw:pseudowires/pw:pseudowire:
    +--rw vccv-ability?     boolean
    +--rw request-vlanid?   uint16
    +--rw vlan-tpid?        string
    +--rw ttl?              uint8
  augment /pw:pseudowires/pw:pseudowire/pw:pw-type:
    +--:(bgp-pw)
    |  +--rw bgp-pw
    |     +--rw remote-pe-id?   inet:ip-address
    +--:(bgp-ad-pw)
       +--rw bgp-ad-pw
          +--rw remote-ve-id?   uint16

  notifications:
    +---n l2vpn-state-change-notification
       +--ro l2vpn-instance-name?   l2vpn-instance-name-ref
       +--ro l2vpn-instance-type?   -> /l2vpn/instances/instance[l2vpn:name=curr
ent()/../l2vpn-instance-name]/type
       +--ro (ac-or-pw-or-redundancy-grp)?
       |  +--:(ac)
```

```
       |  |  +--ro ac?                    -> /l2vpn/instances/instance[name=curr
ent()/../l2vpn-instance-name][type=current()/../l2vpn-instance-type]/endpoint/pw
/name
       |  +--:(pw)
       |  |  +--ro pw?                    -> /l2vpn/instances/instance[name=curr
ent()/../l2vpn-instance-name][type=current()/../l2vpn-instance-type]/endpoint/pw
/name
       |  +--:(redundancy-grp)
       |     +--ro (primary)
       |     |  +--:(primary-ac)
       |     |  |  +--ro primary-ac?            -> /l2vpn/instances/instance[nam
e=current()/../l2vpn-instance-name][type=current()/../l2vpn-instance-type]/endpo
int/primary-ac/name
       |     |  +--:(primary-pw)
       |     |     +--ro primary-pw?            -> /l2vpn/instances/instance[nam
e=current()/../l2vpn-instance-name][type=current()/../l2vpn-instance-type]/endpo
int/primary-pw/name
       |     +--ro (backup)?
       |        +--:(backup-ac)
       |        |  +--ro backup-ac?            -> /l2vpn/instances/instance[nam
e=current()/../l2vpn-instance-name][type=current()/../l2vpn-instance-type]/endpo
int/backup-ac/name
       |        +--:(backup-pw)
       |           +--ro backup-pw?            -> /l2vpn/instances/instance[nam
e=current()/../l2vpn-instance-name][type=current()/../l2vpn-instance-type]/endpo
int/backup-pw/name
       +--ro state?                identityref
```

Figure 2

4. YANG Module

The L2VPN configuration container is logically divided into following high level config areas:

```
<CODE BEGINS> file "ietf-pseudowires@2017-06-26.yang"
module ietf-pseudowires {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pseudowires";
  prefix "pw";

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

  organization  "ietf";
  contact       "ietf";
  description   "Pseudowire YANG model";

  revision "2017-06-26" {
    description "Initial revision " +
                "  - Created a new model for pseudowires, which used " +
                "    to be defined within the L2VPN model " +
                "";
```

```
    reference   "";
  }

  /* Typedefs */

  typedef pseudowire-ref {
    type leafref {
      path "/pw:pseudowires/pw:pseudowire/pw:name";
    }
    description "A type that is a reference to a pseudowire";
  }

  typedef pw-template-ref {
    type leafref {
      path "/pw:pseudowires/pw:pw-templates/pw:pw-template/pw:name";
    }
    description "A type that is a reference to a pw-template";
  }

  typedef cw-negotiation-type {
    type enumeration {
      enum "non-preferred" {
        description "No preference for control-word";
      }
      enum "preferred" {
        description "Prefer to have control-word negotiation";
      }
    }
    description "control-word negotiation preference type";
  }

  typedef pseudowire-status-type {
    type bits {
      bit pseudowire-forwarding {
        position 0;
        description "Pseudowire is forwarding";
      }
      bit pseudowire-not-forwarding {
        position 1;
        description "Pseudowire is not forwarding";
      }
      bit local-attachment-circuit-receive-fault {
        position 2;
        description "Local attachment circuit (ingress) receive " +
                    "fault";
      }
      bit local-attachment-circuit-transmit-fault {
        position 3;
```

```
        description "Local attachment circuit (egress) transmit " +
                    "fault";
      }
      bit local-PSN-facing-PW-receive-fault {
        position 4;
        description "Local PSN-facing PW (ingress) receive fault";
      }
      bit local-PSN-facing-PW-transmit-fault {
        position 5;
        description "Local PSN-facing PW (egress) transmit fault";
      }
      bit PW-preferential-forwarding-status {
        position 6;
        description "Pseudowire preferential forwarding status";
      }
      bit PW-request-switchover-status {
        position 7;
        description "Pseudowire request switchover status";
      }
    }
    description
      "Pseudowire status type, as registered in the IANA " +
      "Pseudowire Status Code Registry";
  }

  /* Groupings */

  grouping pw-type-grp {
    description "pseudowire type grouping";
    choice pw-type {
      description "A choice of pseudowire type";
      case ldp-or-static-pw {
        leaf peer-ip {
          type inet:ip-address;
          description "peer IP address";
        }
        leaf pw-id {
          type uint32;
          description "pseudowire id";
        }
        leaf icb {
          type boolean;
          description "inter-chassis backup";
        }
        leaf transmit-label {
          type rt-types:mpls-label;
          description "transmit lable";
        }
```

```
        leaf receive-label {
          type rt-types:mpls-label;
          description "receive label";
        }
      }
      case bgp-pw {
        leaf remote-pe-id {
          type inet:ip-address;
          description "remote pe id";
        }
      }
      case bgp-ad-pw {
        leaf remote-ve-id {
          type uint16;
          description "remote ve id";
        }
      }
    }
  }

  /* Data */

  container pseudowires {
    description "Configuration management of pseudowires";
    list pseudowire {
      key "name";
      description "A pseudowire";
      leaf name {
        type string;
        description "pseudowire name";
      }
      leaf state {
        type pseudowire-status-type;
        config false;
        description "pseudowire operation status";
        reference "RFC 4446 and IANA Pseudowire Status Codes " +
                  "Registery";
      }
      leaf template {
        type pw-template-ref;
        description "pseudowire template";
      }
      leaf mtu {
        type uint16;
        description "PW MTU";
      }
      leaf mac-withdraw {
        type boolean;
```

```
        default false;
        description "Enable (true) or disable (false) MAC withdraw";
      }
      leaf cw-negotiation {
        type cw-negotiation-type;
        description "cw-negotiation";
      }
      leaf tunnel-policy {
        type string;
        description "tunnel policy name";
      }
      choice pw-type {
        description "A choice of pseudowire type";
        case configured-pw {
          container configured-pw {
            description "LDP or static pseudowire that requires " +
                        "manual configuration";
            leaf peer-ip {
              type inet:ip-address;
              description "peer IP address";
            }
            leaf pw-id {
              type uint32;
              description "pseudowire id";
            }
            leaf icb {
              type boolean;
              description "inter-chassis backup";
            }
            leaf transmit-label {
              type rt-types:mpls-label;
              description "transmit lable";
            }
            leaf receive-label {
              type rt-types:mpls-label;
              description "receive label";
            }
          }
        }
      }
    }
    container pw-templates {
      description "pw-templates";
      list pw-template {
        key "name";
        description "pw-template";
        leaf name {
          type string;
```

```
          description "name";
        }
        leaf mtu {
          type uint16;
          description "pseudowire mtu";
        }
        leaf cw-negotiation {
          type cw-negotiation-type;
          default "preferred";
          description
            "control-word negotiation preference";
        }
        leaf tunnel-policy {
          type string;
          description "tunnel policy name";
        }
      }
    }
  }
}
<CODE ENDS>

<CODE BEGINS> file "ietf-l2vpn@2017-06-26.yang"
module ietf-l2vpn {
  namespace "urn:ietf:params:xml:ns:yang:ietf-l2vpn";
  prefix "l2vpn";

  import ietf-inet-types {
    prefix "inet";
  }

  import ietf-yang-types {
    prefix "yang";
  }

  import ietf-routing-types {
    prefix "rt-types";
  }

  import ietf-pseudowires {
    prefix "pw";
  }

  organization  "ietf";
  contact       "ietf";
  description   "l2vpn";

  revision "2017-06-26" {
    description "Sixth revision " +
```

```
                "  - Removed unused module mpls " +
                "  - Renamed l2vpn-instances-state to l2vpn-instances " +
                "  - Added pseudowire status as defined in RFC4446 and " +
                "    IANA Pseudowire Status Codes Register " +
                "  - Added notifications " +
                "  - Moved PW definition out of L2VPN " +
                "  - Moved model to NMDA style specified in " +
                "    draft-dsdt-nmda-guidelines-01.txt " +
                "  - Renamed l2vpn-instances and l2vpn-instance to " +
                "    instances and instance to shorten xpaths " +
                "";
    reference   "";
  }

  revision "2017-03-06" {
    description "Sixth revision " +
                "  - Removed the 'common' container and move pw-templates " +
                "    and redundancy-group-templates up a level " +
                "  - Consolidated the endpoint configuration such that " +
                "    all L2VPN instances has a list of endpoint.  For " +
                "    certain types of L2VPN instances such as VPWS where " +
                "    each L2VPN instance is limited to at most two " +
                "    endpoint, additional augment statements were included " +
                "    to add necessary constraints " +
                "  - Removed discovery-type and signaling-type operational " +
                "    state from VPLS pseudowires, as these two parameters " +
                "    are configured as L2VPN parameters rather than " +
                "    pseudowire paramteres " +
                "  - Renamed l2vpn-instances to l2vpn-instances-state " +
                "    in the operational state branch " +
                "  - Removed BGP parameter groupings and reused " +
                "    ietf-routing-types.yang module instead " +
                "";
    reference "";
  }

  revision "2016-10-24" {
    description "Fifth revision " +
                "  - Edits based on Giles's comments " +
                "    5) Remove relative leafrefs in groupings, " +
                "       and the resulting new groupings are: " +
                "       (a) bgp-auto-discovery-parameters-grp " +
                "       (b) bgp-signaling-parameters-grp " +
                "       (c) endpoint-grp " +
                "    11) Merge VPLS and VPWS into one single list " +
                "       and use augment statements to handle " +
                "       differences between VPLS and VPWS " +
                "  - Add a new grouping l2vpn-common-parameters-grp " +
```

```
                "    to make VPLS and VPWS more consistent";
    reference "";
  }

  revision "2016-05-31" {
    description "Fourth revision " +
                "  - Edits based on Giles's comments " +
                "    1) Change enumeration to identityref type for: " +
                "       (a) l2vpn-service-type " +
                "       (b) l2vpn-discovery-type " +
                "       (c) l2vpn-signaling-type " +
                "       bgp-rt-type, cw-negotiation, and " +
                "       pbb-component remain enumerations " +
                "    2) Define i-sid-type for leaf 'i-sid' " +
                "       (which is renamed from 'i-tag') " +
                "    3) Rename 'vpn-targets' to 'vpn-target' " +
                "    4) Import ietf-mpls.yang and reuse the " +
                "       'mpls-label' type defined in ietf-mpls.yang " +
                "       transmit-label and receive-label " +
                "    8) Change endpoint list's key to name " +
                "    9) Changed MTU to type uint16 " +
                "";
    reference "";
  }

  revision "2016-03-07" {
    description "Third revision " +
                "  - Changed the module name to ietf-l2vpn " +
                "  - Merged EVPN into L2VPN " +
                "  - Eliminated the definitions of attachment " +
                "    circuit with the intention to reuse other " +
                "    layer-2 definitions " +
                "  - Added state branch";
    reference "";
  }

  revision "2015-10-08" {
    description "Second revision " +
                "  - Added container vpls-instances " +
                "  - Rearranged groupings and typedefs to be " +
                "    reused across vpls-instance and vpws-instances";
    reference "";
  }

  revision "2015-06-30" {
    description "Initial revision";
    reference   "";
  }
```

```
  /* identities */

  identity l2vpn-instance-type {
    description "Base identity from which identities of " +
                "l2vpn service instance types are derived";
  }

  identity vpws-instance-type {
    base l2vpn-instance-type;
    description "This identity represents VPWS instance type";
  }

  identity vpls-instance-type {
    base l2vpn-instance-type;
    description "This identity represents VPLS instance type";
  }

  identity link-discovery-protocol {
    description "Base identiy from which identities describing " +
                "link discovery protocols are derived";
  }

  identity lacp {
    base "link-discovery-protocol";
    description "This identity represents LACP";
  }

  identity lldp {
    base "link-discovery-protocol";
    description "This identity represents LLDP";
  }

  identity bpdu {
    base "link-discovery-protocol";
    description "This identity represens BPDU";
  }

  identity cpd {
    base "link-discovery-protocol";
    description "This identity represents CPD";
  }

  identity udld {
    base "link-discovery-protocol";
    description "This identity represens UDLD";
  }

  identity l2vpn-service {
```

```
    description "Base identity from which identities describing " +
                "L2VPN services are derived";
  }

  identity Ethernet {
    base "l2vpn-service";
    description "This identity represents Ethernet service";
  }

  identity ATM {
    base "l2vpn-service";
    description "This identity represents Asynchronous Transfer " +
                "Mode service";
  }

  identity FR {
    base "l2vpn-service";
    description "This identity represent Frame-Relay service";
  }

  identity TDM {
    base "l2vpn-service";
    description "This identity represent Time Devision " +
                "Multiplexing service";
  }
  identity l2vpn-discovery {
    description "Base identity from which identities describing " +
                "L2VPN discovery protocols are derived";
  }

  identity manual-discovery {
    base "l2vpn-discovery";
    description "Manual configuration of l2vpn service";
  }

  identity bgp-auto-discovery {
    base "l2vpn-discovery";
    description "Border Gateway Protocol (BGP) auto-discovery of " +
                "l2vpn service";
  }

  identity ldp-discovery {
    base "l2vpn-discovery";
    description "Label Distribution Protocol (LDP) discovery of " +
                "l2vpn service";
  }

  identity mixed-discovery {
```

```
    base "l2vpn-discovery";
    description "Mixed discovery methods of l2vpn service";
  }

  identity l2vpn-signaling {
    description "Base identity from which identities describing " +
                "L2VPN signaling protocols are derived";
  }

  identity static-configuration {
    base "l2vpn-signaling";
    description "Static configuration of labels (no signaling)";
  }

  identity ldp-signaling {
    base "l2vpn-signaling";
    description "Label Distribution Protocol (LDP) signaling";
  }

  identity bgp-signaling {
    base "l2vpn-signaling";
    description "Border Gateway Protocol (BGP) signaling";
  }

  identity mixed-signaling {
    base "l2vpn-signaling";
    description "Mixed signaling methods";
  }

  identity l2vpn-notification-state {
    description "The base identity on which notification states " +
                "are based";
  }

  identity MAC-limit-reached {
    base "l2vpn-notification-state";
    description "MAC limit is reached";
  }

  identity MAC-limit-cleared {
    base "l2vpn-notification-state";
    description "MAC limit is cleared";
  }

  identity MTU-mismatched {
    base "l2vpn-notification-state";
    description "MAC is mismatched";
  }
```

```
  identity MTU-mismatched-cleared {
    base "l2vpn-notification-state";
    description "MAC is mismatch is cleared";
  }

  identity state-changed-to-up {
    base "l2vpn-notification-state";
    description "State is changed to UP";
  }

  identity state-changed-to-down {
    base "l2vpn-notification-state";
    description "State is changed to down";
  }

  identity MAC-move-limit-exceeded {
    base "l2vpn-notification-state";
    description "MAC move limit is exceeded";
  }

  identity MAC-move-limit-exceeded-cleared {
    base "l2vpn-notification-state";
    description "MAC move limit exceeded is cleared";
  }

  identity MAC-flap-detected {
    base "l2vpn-notification-state";
    description "MAC flap detected";
  }

  identity port-disabled-due-to-MAC-flap {
    base "l2vpn-notification-state";
    description "Port disabled due to MAC flap";
  }

  /* typedefs */

  typedef l2vpn-service-type {
    type identityref {
      base "l2vpn-service";
    }
    description "L2VPN service type";
  }

  typedef l2vpn-discovery-type {
    type identityref {
      base "l2vpn-discovery";
    }
```

```
    description "L2VPN discovery type";
  }

  typedef l2vpn-signaling-type {
    type identityref {
      base "l2vpn-signaling";
    }
    description "L2VPN signaling type";
  }

  typedef link-discovery-protocol-type {
    type identityref {
      base "link-discovery-protocol";
    }
    description "This type is used to identify " +
                "link discovery protocol";
  }

  typedef pbb-component-type {
    type enumeration {
      enum "b-component" {
        description "Identifies as a b-component";
      }
      enum "i-component" {
        description "Identifies as an i-component";
      }
    }
    description "This type is used to identify " +
                "the type of PBB component";
  }

  typedef redundancy-group-template-ref {
    type leafref {
      path "/l2vpn:l2vpn/l2vpn:redundancy-group-templates" +
           "/l2vpn:redundancy-group-template/l2vpn:name";
    }
    description "redundancy-group-template-ref";
  }

  typedef l2vpn-instance-name-ref {
    type leafref {
      path "/l2vpn:l2vpn/l2vpn:instances" +
           "/l2vpn:instance/l2vpn:name";
    }
    description "l2vpn-instance-name-ref";
  }

  typedef l2vpn-instance-type-ref {
```

```
    type leafref {
      path "/l2vpn:l2vpn/l2vpn:instances" +
           "/l2vpn:instance/l2vpn:type";
    }
    description "l2vpn-instance-type-ref";
  }

  typedef operational-state-type {
    type enumeration {
      enum 'up' {
        description "Operational state is up";
      }
      enum 'down' {
        description "Operational state is down";
      }
    }
    description "operational-state-type";
  }

  typedef i-sid-type {
    type uint32 {
      range "0..16777216";
    }
    description "I-SID type that is 24-bits. " +
                "This should be moved to ieee-types.yang at " +
                "http://www.ieee802.org/1/files/public/docs2015" +
                "/new-mholness-ieee-types-yang-v01.yang";
  }

  /* groupings */

  grouping one-l2vpn-instance-grp {
    description "A grouping that identifies a single L2VPN instance";
    leaf l2vpn-instance-name {
      type l2vpn-instance-name-ref;
      description "The L2VPN instance name";
    }
    leaf l2vpn-instance-type {
      type leafref {
        path "/l2vpn:l2vpn/l2vpn:instances" +
             "/l2vpn:instance" +
                  "[l2vpn:name=current()/../l2vpn-instance-name]" +
             "/l2vpn:type";
      }
      description "The L2VPN instance type";
    }
  }
```

```
  grouping pbb-parameters-grp {
    description "PBB parameters grouping";
    container pbb-parameters {
      description "pbb-parameters";
      choice component-type {
        description "PBB component type";
        case i-component {
          leaf i-sid {
            type i-sid-type;
            description "I-SID";
          }
          leaf backbone-src-mac {
            type yang:mac-address;
            description "backbone-src-mac";
          }
        }
        case b-component {
          leaf bind-b-component-name {
            type l2vpn-instance-name-ref;
            must "/l2vpn:l2vpn" +
                 "/l2vpn:instances" +
                 "/l2vpn:instance[l2vpn:name=current()]" +
                 "/type = 'vpls-instance-type'" {
              description "A b-component must be an L2VPN instance " +
                          "of type vpls-instance-type";
            }
            description "Reference to the associated b-component";
          }
          leaf bind-b-component-type {
            type identityref {
              base l2vpn-instance-type;
            }
            must ". = 'l2vpn:vpls-instance-type'" {
              description "The associated b-component must have " +
                          "type vpls-instance-type";
            }
            config false;
            description "Type of the associated b-component";
          }
        }
      }
    }
  }

  grouping pbb-parameters-state-grp {
    description "PBB parameters grouping";
    container pbb-parameters {
      description "pbb-parameters";
```

```
      choice component-type {
        description "PBB component type";
        case i-component {
          leaf i-sid {
            type i-sid-type;
            description "I-SID";
          }
          leaf backbone-src-mac {
            type yang:mac-address;
            description "backbone-src-mac";
          }
        }
        case b-component {
          leaf bind-b-component-name {
            type string;
            description "Name of the associated b-component";
          }
          leaf bind-b-component-type {
            type identityref {
              base l2vpn-instance-type;
            }
            must ". = 'l2vpn:vpls-instance-type'" {
              description "The associated b-component must have " +
                          "type vpls-instance-type";
            }
            description "Type of the associated b-component";
          }
        }
      }
    }
  }

  grouping l2vpn-common-parameters-grp {
    description "L2VPN common parameters";
    leaf name {
      type string;
      description "Name of L2VPN service instance";
    }
    leaf type {
      type identityref {
        base l2vpn-instance-type;
      }
      description "Type of L2VPN service instance";
    }
    leaf mtu {
      type uint16;
      description "MTU of L2VPN service";
    }
```

```
    leaf mac-aging-timer {
      type uint32;
      description "mac-aging-timer, the duration after which" +
                  "a MAC entry is considered aged out";
    }
    leaf service-type {
      type l2vpn-service-type;
      default Ethernet;
      description "L2VPN service type";
    }
    leaf discovery-type {
      type l2vpn-discovery-type;
      default manual-discovery;
      description "L2VPN service discovery type";
    }
    leaf signaling-type {
      type l2vpn-signaling-type;
      mandatory true;
      description "L2VPN signaling type";
    }
  }

  grouping bgp-signaling-parameters-grp {
    description "BGP parameters for signaling";
    leaf site-id {
      type uint16;
      description "Site ID";
    }
    leaf site-range {
      type uint16;
      description "Site Range";
    }
  }

  grouping redundancy-group-properties-grp {
    description "redundancy-group-properties-grp";
    leaf protection-mode {
      type enumeration {
        enum "frr" {
          value 0;
          description "fast reroute";
        }
        enum "master-slave" {
          value 1;
          description "master-slave";
        }
        enum "independent" {
          value 2;
```

```
          description "independent";
        }
      }
      description "protection-mode";
    }
    leaf reroute-mode {
      type enumeration {
        enum "immediate" {
          value 0;
          description "immediate reroute";
        }
        enum "delayed" {
          value 1;
          description "delayed reroute";
        }
        enum "never" {
          value 2;
          description "never reroute";
        }
      }
      description "reroute-mode";
    }
    leaf dual-receive {
      type boolean;
      description
      "allow extra traffic to be carried by backup";
    }
    leaf revert {
      type boolean;
      description "allow forwarding to revert to primary " +
                  "after restoring primary";
    }
    leaf reroute-delay {
      when "../reroute-mode = 'delayed'" {
        description "Specify amount of time to " +
                    "delay reroute only when " +
                    "delayed route is configured";
      }
      type uint16;
      description "amount of time to delay reroute";
    }
    leaf revert-delay {
      when "../revert = 'true'" {
        description "Specify the amount of time to " +
                    "wait to revert to primary " +
                    "only if reversion is configured";
      }
      type uint16;
```

```
      description "amount ot time to wait to revert to primary";
    }
  }

  grouping endpoint-grp {
    description "A grouping that defines the structure of " +
                "an endpoint";
    choice ac-or-pw-or-redundancy-grp {
      description "A choice ofattachment circuit or " +
                  "pseudowire or redundancy group";
      case ac {
        description "Attachment circuit(s) as an endpoint";
      }
      case pw {
        description "Pseudowire(s) as an endpoint";
      }
      case redundancy-grp {
        description "Redundancy group as an endpoint";
        choice primary {
          mandatory true;
          description "primary options";
          case primary-ac {
            description "primary-ac";
          }
          case primary-pw {
            description "primary-pw";
          }
        }
        choice backup {
          description "backup options";
          case backup-ac {
            description "backup-ac";
          }
          case backup-pw {
            description "backup-pw";
          }
        }
      }
    }
  }

  grouping ac-params-grp {
    description "ac-state-grp";
    leaf name {
      type string;
      description "Name of attachment circuit.  " +
                  "This field is intended to " +
                  "reference standardized " +
```

```
                "layer-2 definitions.";
    }
    leaf state {
      type operational-state-type;
      config false;
      description "attachment circuit up/down state";
    }
  }

  grouping pw-params-grp {
    description "PW state grouping";
    leaf name {
      type pw:pseudowire-ref;
      description "Pseudowire name";
    }
    leaf state {
      type leafref {
        path "/pw:pseudowires" +
             "/pw:pseudowire[pw:name=current()/../name]" +
             "/pw:state";
      }
      config false;
      description "Pseudowire state";
    }
  }

  /* L2VPN YANG Model */

  container l2vpn {
    description "l2vpn";

    container redundancy-group-templates {
      description "redundancy group templates";
      list redundancy-group-template {
        key "name";
        description "redundancy-group-template";
        leaf name {
          type string;
          description "name";
        }
        uses redundancy-group-properties-grp;
      }
    }
    container instances {
      description "A list of L2VPN instances";
      list instance {
        key "name type";
        description "An L2VPN service instance";
```

```
        uses l2vpn-common-parameters-grp;
        container bgp-auto-discovery {
          description "BGP auto-discovery parameters";
          leaf route-distinguisher {
            type rt-types:route-distinguisher;
            description "BGP route distinguisher";
          }
          leaf vpn-id {
            type string;
            description "VPN ID";
          }
          uses rt-types:vpn-route-targets;
        }
        container bgp-signaling {
          when "../signaling-type = 'bgp-signaling'" {
            description "Check signaling type: " +
                        "Can only configure BGP signaling if " +
                        "signaling type is BGP";
          }
          description "BGP signaling parameters";
          uses bgp-signaling-parameters-grp;
        }
        list endpoint {
          key "name";
          description "An endpoint";
          leaf name {
            type string;
            description "endpoint name";
          }
          uses endpoint-grp {
            augment "ac-or-pw-or-redundancy-grp/ac" {
              description "Augment for attachment circuit(s) " +
                          "as an endpoint";
              list ac {
                key "name";
                uses ac-params-grp;
                description "An L2VPN instance's " +
                            "attachment circuit list";
              }
            }
            augment "ac-or-pw-or-redundancy-grp/pw" {
              description "Augment for pseudowire(s) as an endpoint";
              list pw {
                key "name";
                uses pw-params-grp {
                  description "Pseudowire parameters";
                  refine "name" {
                    must "(../../../type = " +
```

```
                    " 'l2vpn:vpws-instance-type') or " +
                    "(not(boolean(/pw:pseudowires" +
                    "     /pw:pseudowire[pw:name = current()]" +
                    "     /vccv-ability)) and " +
                    " not(boolean(/pw:pseudowires" +
                    "     /pw:pseudowire[pw:name = current()]" +
                    "     /request-vlanid)) and " +
                    " not(boolean(/pw:pseudowires" +
                    "     /pw:pseudowire[pw:name = current()]" +
                    "     /vlan-tpid)) and " +
                    " not(boolean(/pw:pseudowires" +
                    "     /pw:pseudowire[pw:name = current()]" +
                    "     /ttl)))" {
                  description "Only a VPWS PW has parameters " +
                              "vccv-ability, request-vlanid, " +
                              "vlan-tpid, and ttl";
                }
              }
            }
            description "An L2VPN instance's pseudowire list";
          }
        }
        augment "ac-or-pw-or-redundancy-grp/redundancy-grp/" +
                "primary/primary-ac" {
          description "Augment for primary-ac";
          container primary-ac {
            description "Primary AC";
            uses ac-params-grp;
          }
        }
        augment "ac-or-pw-or-redundancy-grp/redundancy-grp/" +
                "primary/primary-pw" {
          description "Augment for primary-pw";
          list primary-pw {
            key "name";
            uses pw-params-grp {
              description "Pseudowire parameters";
              refine "name" {
                must "(../../../type = " +
                     " 'l2vpn:vpws-instance-type') or " +
                     "(not(boolean(/pw:pseudowires" +
                     "     /pw:pseudowire[pw:name = current()]" +
                     "     /vccv-ability)) and " +
                     " not(boolean(/pw:pseudowires" +
                     "     /pw:pseudowire[pw:name = current()]" +
                     "     /request-vlanid)) and " +
                     " not(boolean(/pw:pseudowires" +
                     "     /pw:pseudowire[pw:name = current()]" +
```

```
                        "     /vlan-tpid)) and " +
                        " not(boolean(/pw:pseudowires" +
                        "     /pw:pseudowire[pw:name = current()]" +
                        "     /ttl)))" {
                    description "Only a VPWS PW has parameters " +
                                "vccv-ability, request-vlanid, " +
                                "vlan-tpid, and ttl";
                  }
                }
              }
              description "An L2VPN instance's pseudowire list";
            }
          }
          augment "ac-or-pw-or-redundancy-grp/redundancy-grp/" +
                  "backup/backup-ac" {
            description "Augment for backup-ac";
            container backup-ac {
              description "Backup AC";
              uses ac-params-grp;
            }
          }
          augment "ac-or-pw-or-redundancy-grp/redundancy-grp/" +
                  "backup/backup-pw" {
            description "Augment for backup-pw";
            list backup-pw {
              key "name";
              uses pw-params-grp {
                description "Pseudowire parameters";
                refine "name" {
                  must "(../../../type = " +
                       " 'l2vpn:vpws-instance-type') or " +
                       "(not(boolean(/pw:pseudowires" +
                       "     /pw:pseudowire[pw:name = current()]" +
                       "     /vccv-ability)) and " +
                       " not(boolean(/pw:pseudowires" +
                       "     /pw:pseudowire[pw:name = current()]" +
                       "     /request-vlanid)) and " +
                       " not(boolean(/pw:pseudowires" +
                       "     /pw:pseudowire[pw:name = current()]" +
                       "     /vlan-tpid)) and " +
                       " not(boolean(/pw:pseudowires" +
                       "     /pw:pseudowire[pw:name = current()]" +
                       "     /ttl)))" {
                    description "Only a VPWS PW has parameters " +
                                "vccv-ability, request-vlanid, " +
                                "vlan-tpid, and ttl";
                  }
                }
```

```
                }
                description "A list of backup pseudowires";
              }
            }
            augment "ac-or-pw-or-redundancy-grp/redundancy-grp" {
              description "Augment for redundancy group properties";
              leaf template {
                type leafref {
                  path "/l2vpn/redundancy-group-templates" +
                       "/redundancy-group-template/name";
                }
                description "Reference a redundancy group " +
                            "properties template";
              }
              uses redundancy-group-properties-grp;
            }
          }
        }
      }
    }
  }

  /* augments */

  augment "/pw:pseudowires/pw:pseudowire" {
    description "Augment for peudowire parameters for " +
                "VPWS pseudowires";
    leaf vccv-ability {
      type boolean;
      description "vccvability";
    }
    leaf request-vlanid {
      type uint16;
      description "request vlanid";
    }
    leaf vlan-tpid {
      type string;
      description "vlan tpid";
    }
    leaf ttl {
      type uint8;
      description "time-to-live";
    }
  }

  augment "/pw:pseudowires/pw:pseudowire/pw:pw-type" {
    description "Additional pseudowire types";
    case bgp-pw {
```

```
      container bgp-pw {
        description "BGP pseudowire";
        leaf remote-pe-id {
          type inet:ip-address;
          description "remote pe id";
        }
      }
    }
    case bgp-ad-pw {
      container bgp-ad-pw {
        description "BGP auto-discovery pseudowire";
        leaf remote-ve-id {
          type uint16;
          description "remote ve id";
        }
      }
    }
  }

  augment "/l2vpn/instances/instance" {
    when "type = 'l2vpn:vpws-instance-type'" {
      description "Constraints only for VPWS pseudowires";
    }
    description "Augment for VPWS instance";
    container vpws-constraints {
      must "(count(../endpoint) < 2) and " +
           "(count(../endpoint/pw) < 1) and " +
           "(count(../endpoint/ac) < 1) and " +
           "(count(../endpoint/primary-pw) < 1) and " +
           "(count(../endpoint/backup-pw) < 1) " {
        description "A VPWS L2VPN instance has at most 2 endpoints " +
                    "and each endpoint has at most 1 pseudowire or " +
                    "1 attachment circuit";
      }
      description "VPWS constraints";
    }
  }

  augment "/l2vpn/instances/instance" {
    when "type = 'l2vpn:vpls-instance-type'" {
      description "Parameters specifically for a VPLS instance";
    }
    description "Augment for parameters for a VPLS instance";
    uses pbb-parameters-grp;
  }

  augment "/l2vpn/instances/instance/endpoint" {
    when "../type = 'l2vpn:vpls-instance-type'" {
```

```
      description "Endpoint parameter specifically for " +
                  "a VPLS instance";
    }
    description "Augment for endpoint parameters for a VPLS instance";
    leaf split-horizon-group {
      type string;
      description "Identify a split horizon group";
    }
  }

  augment "/l2vpn/instances/instance/endpoint" +
          "/ac-or-pw-or-redundancy-grp/redundancy-grp" +
          "/backup/backup-pw/backup-pw" {
    when "../../type = 'l2vpn:vpls-instance-type'" {
      description "Backup pseudowire parameter specifically for " +
                  "a VPLS instance";
    }
    description "Augment for backup pseudowire paramters for " +
                "a VPLS instance";
    leaf precedence {
      type uint32;
      description "precedence of the pseudowire";
    }
  }

  /* Notifications */

  notification l2vpn-state-change-notification {
    description "L2VPN and constituents state change notification";
    uses one-l2vpn-instance-grp;
    uses endpoint-grp {
      augment "ac-or-pw-or-redundancy-grp/ac" {
        description "Augment for attachment circuit(s) " +
                    "as an endpoint";
        leaf ac {
          type leafref {
            path "/l2vpn/instances/instance" +
                    "[name=current()/../l2vpn-instance-name]" +
                    "[type=current()/../l2vpn-instance-type]" +
                  "/endpoint/pw/name";
          }
          description "Related attachment circuit";
        }
      }
      augment "ac-or-pw-or-redundancy-grp/pw" {
        description "Augment for pseudowire(s) as an endpoint";
        leaf pw {
          type leafref {
```

```
            path "/l2vpn/instances/instance" +
                    "[name=current()/../l2vpn-instance-name]" +
                    "[type=current()/../l2vpn-instance-type]" +
                  "/endpoint/pw/name";
          }
          description "Related pseudowire";
        }
      }
      augment "ac-or-pw-or-redundancy-grp/redundancy-grp/" +
              "primary/primary-ac" {
        description "Augment for primary-ac";
        leaf primary-ac {
          type leafref {
            path "/l2vpn/instances/instance" +
                    "[name=current()/../l2vpn-instance-name]" +
                    "[type=current()/../l2vpn-instance-type]" +
                  "/endpoint/primary-ac/name";
          }
          description "Related primary attachment circuit";
        }
      }
      augment "ac-or-pw-or-redundancy-grp/redundancy-grp/" +
              "primary/primary-pw" {
        description "Augment for primary-pw";
        leaf primary-pw {
          type leafref {
            path "/l2vpn/instances/instance" +
                    "[name=current()/../l2vpn-instance-name]" +
                    "[type=current()/../l2vpn-instance-type]" +
                  "/endpoint/primary-pw/name";
          }
          description "Related primary pseudowire";
        }
      }
      augment "ac-or-pw-or-redundancy-grp/redundancy-grp/" +
              "backup/backup-ac" {
        description "Augment for backup-ac";
        leaf backup-ac {
          type leafref {
            path "/l2vpn/instances/instance" +
                    "[name=current()/../l2vpn-instance-name]" +
                    "[type=current()/../l2vpn-instance-type]" +
                  "/endpoint/backup-ac/name";
          }
          description "Related backup attachment circuit";
        }
      }
      augment "ac-or-pw-or-redundancy-grp/redundancy-grp/" +
```

```
              "backup/backup-pw" {
          description "Augment for backup-pw";
          leaf backup-pw {
            type leafref {
              path "/l2vpn/instances/instance" +
                      "[name=current()/../l2vpn-instance-name]" +
                      "[type=current()/../l2vpn-instance-type]" +
                    "/endpoint/backup-pw/name";
            }
            description "Related backup pseudowire";
          }
        }
      }
      leaf state {
        type identityref {
          base l2vpn-notification-state;
        }
        description "State change notification";
      }
    }

}


<CODE ENDS>
```

Figure 3

5. Security Considerations

The configuration, state, action and notification data defined in this document are designed to be accessed via the NETCONF protocol [RFC6241]. The lowest NETCONF layer is the secure transport layer and the mandatory-to-implement secure transport is SSH [RFC6242]. The NETCONF access control model [RFC6536] provides means to restrict access for particular NETCONF users to a pre-configured subset of all available NETCONF protocol operations and content.

The security concerns listed above are, however, no different than faced by other routing protocols. Hence, this draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg]

6. IANA Considerations

None.

7. Acknowledgments

The authors would like to acknowledge Giles Heron and others for their useful comments.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

8.2. Informative References

[RFC3916] Xiao, X., Ed., McPherson, D., Ed., and P. Pate, Ed., "Requirements for Pseudo-Wire Emulation Edge-to-Edge (PWE3)", RFC 3916, DOI 10.17487/RFC3916, September 2004, <http://www.rfc-editor.org/info/rfc3916>.

[RFC3985] Bryant, S., Ed. and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, DOI 10.17487/RFC3985, March 2005, <http://www.rfc-editor.org/info/rfc3985>.

[RFC4385] Bryant, S., Swallow, G., Martini, L., and D. McPherson, "Pseudowire Emulation Edge-to-Edge (PWE3) Control Word for Use over an MPLS PSN", RFC 4385, DOI 10.17487/RFC4385, February 2006, <http://www.rfc-editor.org/info/rfc4385>.

[RFC4446] Martini, L., "IANA Allocations for Pseudowire Edge to Edge Emulation (PWE3)", BCP 116, RFC 4446, DOI 10.17487/RFC4446, April 2006, <http://www.rfc-editor.org/info/rfc4446>.

[RFC4447] Martini, L., Ed., Rosen, E., El-Aawar, N., Smith, T., and G. Heron, "Pseudowire Setup and Maintenance Using the Label Distribution Protocol (LDP)", RFC 4447, DOI 10.17487/RFC4447, April 2006, <http://www.rfc-editor.org/info/rfc4447>.

[RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, DOI 10.17487/RFC4448, April 2006, <http://www.rfc-editor.org/info/rfc4448>.

[RFC4664] Andersson, L., Ed. and E. Rosen, Ed., "Framework for Layer 2 Virtual Private Networks (L2VPNs)", RFC 4664, DOI 10.17487/RFC4664, September 2006, <http://www.rfc-editor.org/info/rfc4664>.

[RFC4665] Augustyn, W., Ed. and Y. Serbest, Ed., "Service Requirements for Layer 2 Provider-Provisioned Virtual Private Networks", RFC 4665, DOI 10.17487/RFC4665, September 2006, <http://www.rfc-editor.org/info/rfc4665>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <http://www.rfc-editor.org/info/rfc4761>.

[RFC4762] Lasserre, M., Ed. and V. Kompella, Ed., "Virtual Private LAN Service (VPLS) Using Label Distribution Protocol (LDP) Signaling", RFC 4762, DOI 10.17487/RFC4762, January 2007, <http://www.rfc-editor.org/info/rfc4762>.

[RFC5003] Metz, C., Martini, L., Balus, F., and J. Sugimoto, "Attachment Individual Identifier (AII) Types for Aggregation", RFC 5003, DOI 10.17487/RFC5003, September 2007, <http://www.rfc-editor.org/info/rfc5003>.

[RFC5254] Bitar, N., Ed., Bocci, M., Ed., and L. Martini, Ed., "Requirements for Multi-Segment Pseudowire Emulation Edge-to-Edge (PWE3)", RFC 5254, DOI 10.17487/RFC5254, October 2008, <http://www.rfc-editor.org/info/rfc5254>.

[RFC5659] Bocci, M. and S. Bryant, "An Architecture for Multi-Segment Pseudowire Emulation Edge-to-Edge", RFC 5659, DOI 10.17487/RFC5659, October 2009, <http://www.rfc-editor.org/info/rfc5659>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6073] Martini, L., Metz, C., Nadeau, T., Bocci, M., and M. Aissaoui, "Segmented Pseudowire", RFC 6073, DOI 10.17487/RFC6073, January 2011, <http://www.rfc-editor.org/info/rfc6073>.

[RFC6074] Rosen, E., Davie, B., Radoaca, V., and W. Luo, "Provisioning, Auto-Discovery, and Signaling in Layer 2 Virtual Private Networks (L2VPNs)", RFC 6074, DOI 10.17487/RFC6074, January 2011, <http://www.rfc-editor.org/info/rfc6074>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC6391] Bryant, S., Ed., Filsfils, C., Drafz, U., Kompella, V., Regan, J., and S. Amante, "Flow-Aware Transport of Pseudowires over an MPLS Packet Switched Network", RFC 6391, DOI 10.17487/RFC6391, November 2011, <http://www.rfc-editor.org/info/rfc6391>.

[RFC6423] Li, H., Martini, L., He, J., and F. Huang, "Using the Generic Associated Channel Label for Pseudowire in the MPLS Transport Profile (MPLS-TP)", RFC 6423, DOI 10.17487/RFC6423, November 2011, <http://www.rfc-editor.org/info/rfc6423>.

[RFC6478] Martini, L., Swallow, G., Heron, G., and M. Bocci, "Pseudowire Status for Static Pseudowires", RFC 6478, DOI 10.17487/RFC6478, May 2012, <http://www.rfc-editor.org/info/rfc6478>.

[RFC6536] Bierman, A. and M. Bjorklund, "Network Configuration Protocol (NETCONF) Access Control Model", RFC 6536, DOI 10.17487/RFC6536, March 2012, <http://www.rfc-editor.org/info/rfc6536>.

[RFC6624] Kompella, K., Kothari, B., and R. Cherukuri, "Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling", RFC 6624, DOI 10.17487/RFC6624, May 2012, <http://www.rfc-editor.org/info/rfc6624>.

[RFC7041] Balus, F., Ed., Sajassi, A., Ed., and N. Bitar, Ed., "Extensions to the Virtual Private LAN Service (VPLS) Provider Edge (PE) Model for Provider Backbone Bridging", RFC 7041, DOI 10.17487/RFC7041, November 2013, <http://www.rfc-editor.org/info/rfc7041>.

[RFC7361] Dutta, P., Balus, F., Stokes, O., Calvignac, G., and D. Fedyk, "LDP Extensions for Optimized MAC Address Withdrawal in a Hierarchical Virtual Private LAN Service (H-VPLS)", RFC 7361, DOI 10.17487/RFC7361, September 2014, <http://www.rfc-editor.org/info/rfc7361>.

Appendix A. Example Configuration

This section shows an example configuration using the YANG data model defined in the document.

Appendix B. Contributors

The editors gratefully acknowledge the following people for their contributions to this document.

Reshad Rahman
Cisco Systems, Inc.
Email: rrahman@cisco.com

Kamran Raza
Cisco Systems, Inc.
Email: skraza@cisco.com

Giles Heron
Cisco Systems, Inc.
Email: giheron@cisco.com

Tapraj Singh
Cisco Systems, Inc.
Email: tsingh@cisco.com

Zhenbin Li
Huawei Technologies
Email: lizhenbin@huawei.com

Zhuang Shunwan
Huawei Technologies
Email: Zhuangshunwan@huawei.com

Wang Haibo
Huawei Technologies

Email: rainsword.wang@huawei.com

Sajjad Ahmed
Ericsson
Email: sajjad.ahmed@ericsson.com

Matthew Bocci
Nokia
Email: matthew.bocci@nokia.com

Jorge Rabadan
Nokia
Email: jorge.rabadan@nokia.com

Jonathan Hardwick
Metaswitch
Email: jonathan.hardwick@metaswitch.com

Santosh Esale
Juniper Networks
Email: sesale@juniper.net

Nick Delregno
Verizon
Email: nick.deregno@verizon.com

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

Maria Joecylyn
Verizon
Email: joecylyn.malit@verizon.com

Figure 4

Authors' Addresses

Himanshu Shah
Ciena Corporation

Email: hshah@ciena.com

Patrice Brissette
Cisco Systems, Inc.

Email: pbrisset@cisco.com

Ing-When Chen
Jabil

Email: ing-wher_chen@jabil.com

Iftekar Hussain
Infinera Corporation

Email: ihussain@infinera.com

Bin Wen
Comcast

Email: Bin_Wen@cable.comcast.com

Kishore Tiruveedhula
Juniper Networks

Email: kishoret@juniper.net

L2VPNs
Internet-Draft
Intended status: Standards Track
Expires: September 12, 2017

K. Patel
Arrcus
A. Sajassi
Cisco Systems
J. Drake
Z. Zhang
Juniper Networks, Inc.
W. Henderickx
Nokia

March 12, 2017

Virtual Hub-and-Spoke in BGP EVPNs
draft-keyupate-bess-evpn-virtual-hub-00

Abstract

Ethernet Virtual Private Network (EVPN) solution is becoming pervasive for Network Virtualization Overlay (NVO) services in data center (DC) applications and as the next generation virtual private LAN services in service provider (SP) applications.

The use of host IP default route and host unknown MAC route within a DC is well understood in order to ensure that leaf nodes within a DC only learn and store host MAC and IP addresses for that DC. All other host MAC and IP addresses from remote DCs are learned and stored in DC GW nodes thus alleviating leaf nodes from learning host MAC and IP addresses from the remote DCs.

This draft further optimizes the MAC and IP address learning at the leaf nodes such that a leaf node within a DC only needs to learn and store MAC and IP addresses associated with the sites directly connected to it. A leaf node does not need to learn and store MAC and IP addresses from any other leaf nodes thus reducing the number of learned MACs and IP addresses per EVI substantially.

The modifications provided by this draft updates and extends RFC7024 for BGP EVPN Address Family.

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 http://datatracker.ietf.org/drafts/current/.

Copyright Notice

Table of Contents

1. Introduction 3
2. Requirements Language 4
3. Terminology 4
4. Routing Information Exchange for EVPN routes 5
5. EVPN unknown MAC Route 5
  5.1. Originating EVPN Unknown MAC Route by a V-Hub 5
  5.2. Processing VPN-MAC EVPN unknown Route by a V-SPOKE . . . 5
  5.3. Aliasing 6
  5.4. Split-Horizon And Mass Withdraw 7
6. Forwarding Considerations 7
  6.1. IP-only Forwarding 7
  6.2. MAC-only Forwarding - Bridging 7
  6.3. MAC and IP Forwarding - IRB 8
7. Handling of the Broadcast and Multicast traffic 8
  7.1. Split Horizon 9
  7.2. Route Advertisement 9
  7.3. Designated Forwarder in a Cluster 10
  7.4. Traffic Forwarding Rules 10
    7.4.1. Traffic from Local ACs 10
    7.4.2. Traffic Received by a V-hub from Another PE 11
    7.4.3. Traffic received by a V-spoke from a V-hub 11
  7.5. Multi-homing support 11
  7.6. Direct V-spoke to V-spoke traffic 12
8. ARP/ND Suppression 12

9. IANA Considerations 13
10. Security Considerations 13
11. Acknowledgements 13
12. Change Log 13
13. References 13
13.1. Normative References 13
13.2. Informative References 15
Authors' Addresses 15

1. Introduction

Ethernet Virtual Private Network (EVPN) solution is becoming pervasive for Network Virtualization Overlay (NVO) services in data center (DC) applications and as the next generation virtual private LAN services in service provider (SP) applications.

With EVPN, providing any-to-any connectivity among sites of a given EVPN Instance (EVI) would require each Provider Edge (PE) router connected to one or more of these sites to hold all the host MAC and IP addresses for that EVI. The use of host IP default route and host unknown MAC route within a DC is well understood in order to alleviate the learning of host MAC and IP addresses to only leaf nodes (PEs) within that DC. All other host MAC and IP addresses from remote DCs are learned and stored in DC GW nodes thus alleviating leaf nodes from learning host MAC and IP addresses from the remote DCs.

This draft further optimizes the MAC and IP address learning at the leaf nodes such that a leaf node within a DC only needs to learn and store MAC and IP addresses associated with the sites directly connected to it. A leaf node does not need to learn and store MAC and IP addresses from any other leaf nodes thus reducing the number of learned MACs and IP addresses per EVI substantially.

[RFC7024] provides rules for Hub and Spoke VPNs for BGP L3VPNs. This draft updates and extends [RFC7024] for BGP EVPN Address Family. This draft provides rules for Originating and Processing of the EVPN host unknown MAC route and host default IP route by EVPN Virtual Hub (V-HUB). This draft also provides rules for the handling of the BUM traffic in Hub and Spoke EVPNs and handling of ARP suppression.

The leaf nodes and DC GW nodes in a data center are referred to as Virtual Spokes (V-spokes) and Virtual Hubs (V-hubs) respectively. A set of V-spoke can be associated with one or more V-hubs. If a V-spokes is associated with more than one V-hubs, then it can load balanced traffic among these V-hubs. Different V-spokes can be associated with different sets of V-hubs such that at one extreme each V-spoke can have a different V-hub set although this may not be

desirable and a more typical scenario may be to associate a set of V-spokes to a set of V-hubs - e.g., topology for a DC POD where a set of V-spokes are associated with a set of spine nodes or DC GW nodes.

In order to avoid repeating many of the materials covered in [RFC7024], this draft is written as a delta document with its sections organized to follow those of that RFC with only delta description pertinent to EVPN operation in each section. Therefore, it is assumed that the readers are very familiar with [RFC7024] and EVPN.

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [RFC2119] only when they appear in all upper case. They may also appear in lower or mixed case as English words, without any normative meaning.

3. Terminology

ARP: Address Resolution Protocol
BEB: Backbone Edge Bridge
B-MAC: Backbone MAC Address
CE: Customer Edge
C-MAC: Customer/Client MAC Address
ES: Ethernet Segment
ESI: Ethernet Segment Identifier
IRB: Integrated Routing and Bridging
LSP: Label Switched Path
MP2MP: Multipoint to Multipoint
MP2P: Multipoint to Point
ND: Neighbor Discovery
NA: Neighbor Advertisement
P2MP: Point to Multipoint
P2P: Point to Point
PE: Provider Edge
EVPN: Ethernet VPN
EVI: EVPN Instance
RT: Route Target

Single-Active Redundancy Mode: When only a single PE, among a group of PEs attached to an Ethernet segment, is allowed to forward traffic to/from that Ethernet Segment, then the Ethernet segment is defined to be operating in Single-Active redundancy mode.

All-Active Redundancy Mode: When all PEs attached to an Ethernet segment are allowed to forward traffic to/from that Ethernet Segment,

then the Ethernet segment is defined to be operating in All-Active redundancy mode.

4. Routing Information Exchange for EVPN routes

[RFC7024] defines multiple Route Types NLRI along with procedures for advertisements and processing of these routes. Some of these procedures are impacted as the result of hub-and-spoke architecture. The routing information exchange among the hub, spoke, and vanilla PEs are subject to the same rules as described in section 3 of [RFC7024]. Furthermore, if there are any changes to the EVPN route advisements and processing from advertisements and processing from [RFC7024], they are described below.

5. EVPN unknown MAC Route

Section 3 of [RFC7024] talks about how a V-hub of a given VPN must export a VPN-IP default route for that VPN and this route must be exported to only the V-spokes of that VPN associated with that V-hub. [I-D.EVPN-overlay] defines the notion of the unknown MAC route for an EVI which is analogous to a VPN-IP default route for a VPN. This unknown MAC route is exported by a V-hub to its associated V-spokes. If multiple V-hubs are associated with a set of V-spokes, then each V- hub advertises it with a distinct RD when originating this route. If a V-spoke imports several of these unknown MAC routes and they all have the same preference, then traffic from the V-spoke to other sites of that EVI would be load balanced among the V-hubs.

5.1. Originating EVPN Unknown MAC Route by a V-Hub

Section 7.3 of the [RFC7024] defines procedures for originating a VPN-IP default route for a VPN. The same procuedures apply when a V-hub wants to originate EVPN unknown MAC route for a given EVI. The V-hub MUST announce unknown MAC route using the MAC/IP advertisement route along with the Default Gateway extended community as defined in section 10.1 of the [RFC7432].

5.2. Processing VPN-MAC EVPN unknown Route by a V-SPOKE

Within a given EVPN, a V-spoke MUST import all the unknown MAC routes unless the route-target mismatch happens. The processing of the received VPN-MAC EVPN default route follows the rules explained in the section 3 of the [RFC7024]. The unknown MAC route MUST be installed according to the rules of MAC/IP Advertisement route installation rules in section 9.2.2 of [RFC7024].

In absense of any more specific VPN-MAC EVPN routes, V-spokes installing the unknown MAC route MUST use the route when performing

ARP proxy. This behavior would allow V-Spokes to forward the traffic towards V-Hub.

5.3. Aliasing

[RFC7432] describes the concept and procedures for Aliasing where a station is multi-homed to multiple PEs operating in an All-Active redundancy mode, it is possible that only a single PE learns a set of MAC addresses associated with traffic transmitted by the station. [RFC7432] describes the concepts and procedures for Aliasing, which occurs when a CE is multi-homed to multiple PE nodes, operating in all-active redundancy mode, but not all of the PEs learn the CE's set of MAC addresses. This leads to a situation where remote PEs receive MAC advertisement routes, for these addresses, from a single NVE even though multiple NVEs are connected to the multi-homed station. As a result, the remote NVEs are not able to effectively load-balance traffic among the NVEs connected to the multi-homed Ethernet segment.

To alleviate this issue, EVPN introduces the concept of Aliasing. This refers to the ability of a PE to signal that it has reachability to a given locally attached Ethernet segment, even when it has learnt no MAC addresses from that segment. The Ethernet A-D per-EVI route is used to that end. Remote PEs which receive MAC advertisement routes with non-zero ESI SHOULD consider the MAC address as reachable via all NVEs that advertise reachability to the relevant Segment using Ethernet A-D routes with the same ESI and with the Single-Active flag reset.

This procedure is impacted for virtual hub-and-spoke topology because a given V-spoke does not receive any MAC/IP advertisements from remote V-spokes; therefore, there is no point in propagating Ethernet A-D per-EVI route to the remote V-spokes. In this solution, the V-hubs terminate the Ethernet A-D per-EVI route (used for Aliasing) and follows the procedures described in [RFC7432] for handling this route.

There are scenarios for which it is desirable to establish direct communication path between a pair of V-spokes for a given host MAC address. In such scenario, the advertising V-spoke advertises both the MAC/IP route and Ethernet A-D per-EVI route with the RT of V-hub (RT-VH) per section 3 of [RFC7024]. The use of RT-VH, ensures that these routes are received by the V-spokes associated with that V-hub set and thus enables the V-spokes to perform the Aliasing procedure.

In summary, PE devices (V-hubs in general and V-spokes occasionally) that receive EVPN MAC/IP route advertisements (associated with a multi-homed site) need to also receive the associated Ethernet A-D

per-EVI route advertisement(s) in order for them to perform Aliasing procedure.

5.4. Split-Horizon And Mass Withdraw

[RFC7432] uses Ethernet A-D per-ES route to a) signal to remote PEs the multi-homing redundancy type (Single-Active versus All-Active), b) advertise ESI label for split-horizon filtering when MPLS encapsulation is used, and c) advertise mass-withdraw when a failure of an access interface impacts many MAC addresses. This route does not need to be advertise from a V-spoke to any remote V-spoke unless a direct communication path between a pair of spoke is needed for a given flow.

Even if communication between a pair of V-spoke is needed for just a single flow, the Ethernet A-D per ES route needs to be advertised from the originating V-spoke for that ES which may handle tens or hundreds of thousands of flows. This is because in order to perform Aliasing function for a given flow, the Ethernet A-D per-EVI route is needed and this route itself is dependent on the Ethernet A-D per-ES route. In such scenario, the advertising V-spoke advertises the Ethernet A-D per-ES route with the RT of V-hub (RT-VH) per section 3 of [RFC7024].

In summary, PE devices (V-hubs in general and V-spokes occasionally) that receive EVPN MAC/IP route advertisements (associated with a multi-homed site) need to also receive the associated Ethernet A-D per-ES route advertisement(s).

6. Forwarding Considerations

6.1. IP-only Forwarding

When EVPN operates in IP-only forwarding mode using EVPN Route Type 5, then all forwarding considerations in section 4 of [RFC7024] are directly applicable here.

6.2. MAC-only Forwarding - Bridging

When EVPN operates in MAC-only forwarding mode (i.e., bridging mode), then for a given EVI, the MPLS label that a V-hub advertises with an Unknown MAC address MUST be the label that identifies the MAC-VRF of the V-hub in absense of a more specific MAC route. When the V-hub receives a packet with such label, the V- hub pops the label and determines further disposition of the packet based on the lookup in the MAC-VRF. Otherwise, the MPLS label of the matching more specific route is used and packet is is forwarded towards the associated NEXTHOP of the more specific route.

6.3. MAC and IP Forwarding - IRB

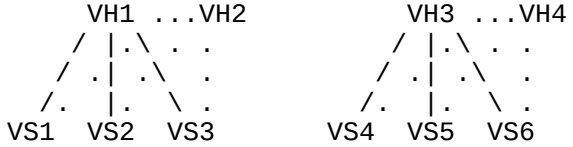
When a EVPN speaker operates in IRB mode, it implements both the a€œIP and MAC forwarding Modesa€ (aka Integrated Routing and Bridging - IRB). On a packet by packet basis, the V-spoke decides whether to do forwarding based on a MAC address lookup (bridge) or based on a IP address lookup (route). If the host destination MAC address is that of the IRB interface (i.e., if the traffic is inter-subnet), then the V-spoke performs an additional IP lookup in the IP-VRF. However, if the host destination MAC address is that of an actual host MAC address (i.e., the traffic is intra-subnet) , then the V-spoke only performs a MAC lookup in the MAC-VRF. The procedure specified in Section 6.1 and Section 6.2 are applicable to inter-subnet and intra-subnet forwarding respectively. For intra-subnet traffic, if the MAC address is not found in the MAC-VRF, then the V-spoke forwards the traffic to the V-hub with the MPLS label received from the V-hub for the unknown MAC address. For the Inter-subnet traffic, if the IP prefix is not found in the IP-VRF, then the V-spoke forwards the traffic to the V-hub with the MPLS label received from the V-hub for the default IP address.

7. Handling of the Broadcast and Multicast traffic

Just like that V-spoke to V-spoke known unicast traffic is relayed by V-hubs, V-spoke to V-spoke BUM traffic can also relayed by V-hubs. This is especially desired if Ingress Replication (IR) would be used otherwise for V-spokes to send traffic to other V-spokes. This way, a V-spoke can unicast BUM traffic to a single V-hub, who will then relay the traffic. This achieves Assisted Replication, and reduces multicast state in the core. Note that a V-hub may relay traffic using MPLS P2MP tunnels or BIER as well as IR. While a V-spoke may use P2MP tunnels or BIER to send traffic to V-hubs, this specification focuses on using IR by V-spokes.

In this particular section, all traffic refers to BUM traffic unless explicitly stated otherwise. The term PE refers to a V-hub or V-spoke when there is no need to distinguish the two.

Consider the following topology, where V-spokes VS1/2/3 are associated with V-hubs VH1/2 in one cluster, and V-spokes VS4/5/6 are associated with V-hubs VH3/4 in another cluster. Note that the lines/dots in the diagram indcate association, not connection.



7.1. Split Horizon

When VH1 relays traffic that it receives from VS1, in case of IR it MUST not send traffic back to VS1, and in case of P2MP tunnel it must indicate that traffic is sourced from VS1 so that VS1 will discard the traffic. In case of IR with IP unicsat tunnels, the outer source IP address identifies the sending PE. In case of IR with MPLS unicast tunnels, VH1 must advertise different labels to different PEs, so that it can identify the sending PE based on the label in the traffic from a V-spoke.

If MPLS P2MP/multicast tunnels (including VXLAN-GPE and MPLS-over-GRE/UDP) are used by a V-hub to relay traffic, an upstream allocated (by the V-hub) label MUST be imposed in the label stack to identify the source of the V-spoke. The label is advertised as part of the PE Distinguisher (PED) Label Attribute of the Inclusive Multicast Ethernet Tag (IMET) route from the V-hub, as specified in Section 8 of [RFC 6514].

Notice that an "upstream-assigned" label used by a V-hub to send traffic with on a P2MP tunnel to identify the source V-spoke is the same "downstream-assigned" label used by the V-hub to receive traffic on the IR tunnel from the V-spoke. Therefore, the same PED Label attribute serves two purposes. With [RFC 6514], a PED label may only identify a PE but not a particular VPN. Here the PED label identifies both the PE and a particular EVI/BD. A V-spoke programs its context MPLS forwarding table for the V-hub to discard any traffic with the PED label that the V-hub advertised for this V-spoke, or pop other PED labels and direct traffic into a corresponding EVI for L2 forwarding.

Note that a V-hub cannot use VXLAN/NVGRE multicast tunnels to relay traffic because if the V-hub uses the source V-spoke’s IP address in the outer IP header (for the purpose of identifying the source V-spoke), multicast RPF would fail and the packets will be discarded.

7.2. Route Advertisement

As with other route types, IMET routes from V-hubs are advertised with RT-VH and RT-EVI so they are imported by associated V-spokes and all V-hubs. They carry the PED Label attribute as described above.

IMET routes from V-spokes are advertised with RT-EVI so they are imported by all V-hubs. They also carry PED Label attribute for multi-homing split horizon purpose if and only if V-hubs uses IR to relay traffic.

If a V-hub uses RSVP-TE P2MP tunnel, IR, or BIER to send or relay traffic, all other PEs (V-hubs or V-spokes) will receive traffic directly because the V-hub sees all PEs. If a V-hub uses mLDP P2MP tunnel to send or relay traffic, only its associated V-spokes and all V-hubs will see the V-hub's IMET route and join the tunnel announced in the route. Another V-hub need to relay traffic to its associated V-spokes that are not associated with this V-hub.

For that V-hub to announce the mLDP relay tunnel in its cluster, it needs to advertise a (*,*) S-PMSI AD route, as specified in [draft-zzhang-bess-evpn-bum-procedure-updates]. The route is advertised with the RT-VH for that cluster, and associated V-spokes will join the tunnel announced in the S-SPMI AD route.

7.3. Designated Forwarder in a Cluster

When there are multiple V-hubs in a cluster, a V-spoke in that cluster decides by itself to which V-hub to send traffic. If the receiving V-hub uses mLDP tunnel to relay traffic, V-hubs in other clusters need to further relay traffic, but only one V-hub in each cluster can do so. As a result, a DF must be elected among the V-hubs for each cluster.

The election is similar to DF election in RFC 7432, with the folllowing differences.

- Instead of using Ethernet Segment route to discover the PEs on a multi-homing ES, the IMET route are used to determine the V-hubs in the same cluster - they all carry the same pair of RT-EVI and RT-VH, and advertises the unknown mac route.

- Instead of using VLAN to do per-VLAN DF election, the Local Administration Field of the RT-EVI is used to do per-EVI DF election.

7.4. Traffic Forwarding Rules

When a PE needs to forward received traffic from local Attachment Circuits (ACs) or remote PEs to local ACs, it follows the rules in RFC 7432, except that traffic sourced from this local PE but relayed back on a p2mp tunnel is discarded. It may also need to forward to other PEs, subject to rules in the following sections.

7.4.1. Traffic from Local ACs

Traffic from a V-hub's local ACs is forwarded using the tunnel announced in its IMET route, as specified in RFC 7432. In case of an mLDP tunnel, the traffic need to be relayed by V-hubs of other

clusters to their associated V-spokes. For other tunnel types, no relay is needed.

Traffic from a V-spoke's local ACs is forwarded to an associated V-hub of its choice. In case of MPLS IR, the label in the V-hub's IMET route's PED attribute corresponding to this V-spoke is used.

7.4.2. Traffic Received by a V-hub from Another PE

When a V-hub receives traffic from an associated V-spoke, it needs to relay to other PEs, using the tunnel announced in its IMET route. In case of IR or BIER, the source V-spoke, which is determined from the incoming label or source IP address, is excluded from the replication list. In case of a P2MP tunnel, the popped incoming label is imposed again to identify the source PE, before the tunnel label is imposed.

When a V-hub receives traffic from another V-hub on a P2MP tunnel, and the tunnel is announced in an IMET route carrying the same RT-VH as this V-hub is configured with, it does not need to relay the traffic. Otherwise, the traffic is from a V-hub in a different cluster, and this V-hub needs to relay to its associated V-spokes, if and only if it is the DF for this cluster, using the tunnel announced in its (*,*) S-PMSI route carrying its RT-VH.

When a V-hub recevies traffic from another V-hub via IR or BIER, it does not further relay the traffic as that V-hub can reach all PEs.

7.4.3. Traffic received by a V-spoke from a V-hub

In case of P2MP tunnel, the V-spoke discards the traffic if the label following the tunnel label identifies the V-spoke itself.

7.5. Multi-homing support

Consider that an ES spans across two V-spokes in the same cluster and the V-hub uses MPLS IR to relay traffic. With ESI Label split horizon method, a source V-spoke uses the ESI label advertised by the V-hub for the ES, and the V-hub must change that to the ESI label advertised by receiving v-spokes when it relays traffic. That means V-hubs must advertise ESI labels for all multi-homing segments, even when they're not on those segments. They must also do double label swap (EVI/BD label and ESI label) or mac lookup when relaying traffic.

To avoid that complexity, Local Bias is the preferred method for split horizon. The PED label following the mpls transport tunnel label or BIER header identifies the PE that originated the traffic in addition to identifying the EVI/BD.

If a V-hub uses P2MP or BIER to relay traffic, the PED label is one of the labels in the PE Distinguiser Label attribute in the V-hub's IMET route, allocated by the V-hub for the source V-spoke.

If a V-hub uses IR to relay traffic, for each V-spoke that it relays to, the PED label advertised by that receiving V-spoke for the source V-spoke needs to be imposed by the V-hub. For that purpose, each V-spoke must include the PED Label attribute in its IMET route, to advertise different labels for different PEs. It discovers the PEs that it needs to advertise labels for via the PED label Attribute in the V-hub's IMET route.

7.6. Direct V-spoke to V-spoke traffic

It may be desired for allow direct V-spoke to V-spoke traffic in a cluster, without the relay by a V-hub.

To do that, V-spokes advertise their IMET routes with both RT-VH and RT-EVI.

Forwarding rules will be specified in future revisions.

8. ARP/ND Suppression

[RFC7432] defines the procedures for ARP/ND suppression where a PE can terminate gratuitous ARP/ND request message from directly connected site and advertises the associated MAC and IP addresses in an EVPN MAC/IP advertisement route to all other remote PEs. The remote PEs that receive this EVPN route advertisement, install the MAC/IP pair in their ARP/ND cache table thus enabling them to terminate ARP/ND requests and generate ARP/ND responses locally thus suppressing the flooding of ARP/ND requests over the EVPN network.

In this hub-and-spoke approach, the ARP suppression needs to be performed by both the EVPN V-hubs as well V-spokes as follow. When a V-Spoke receives a gratuitous ARP/ND request, it terminates it and stores the source MAC/IP pair in its ARP/ND cache table. Then, it advertises the source MAC/IP pair to its associated V-Hubs using EVPN MAC/IP advertisement route. The V-Hubs upon receiving this EVPN route advertisement, create an entry in their ARP/ND cache table for this MAC/IP pair.

Now when a V-Spoke receives an ARP/ND request, it first looks up its ARP cache table, if an entry for that MAC/IP pair is found, then an ARP/ND response is generated locally and sent to the CE. However, if an entry is not found, then the ARP/ND request is unicasted to one of the V-hub associated with this V-spoke. Since, the associated V-hub keeps all the MAC/IP ARP entries in its cache table, it can formulate

and ARP/ND response and forward it to that CE via the corresponding V-spoke.

9. IANA Considerations

This document does NOT make any new requests for IANA allocations.

10. Security Considerations

All the security considerations in [RFC7432] apply directly to this document because this document leverages [RFC7432] control plane and their associated procedures - although not the complete set but rather a subset.

This draft does not introduce any new security considerations beyond that of [RFC7432] and [RFC4761] because advertisements and processing of B-MAC addresses follow that of [RFC7432] and processing of C-MAC addresses follow that of [RFC4761] - i.e, B-MAC addresses are learned in control plane and C-MAC addresses are learned in data plane.

11. Acknowledgements

The authors would like to thank Yakov Rekhter for initial idea discussions.

12. Change Log

Initial Version: Sep 21 2014

13. References

13.1. Normative References

[I-D.zzhang-bess-evpn-bum-procedure-updates]
Zhang, J., Lin, W., Rabadan, J., and K. Patel, "Updates on EVPN BUM Procedures", draft-zzhang-bess-evpn-bum-procedure-updates-01 (work in progress), December 2015.

[RFC1771] Rekhter, Y. and T. Li, "A Border Gateway Protocol 4 (BGP-4)", RFC 1771, DOI 10.17487/RFC1771, March 1995, <http://www.rfc-editor.org/info/rfc1771>.

[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>.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, DOI 10.17487/RFC2784, March 2000, <http://www.rfc-editor.org/info/rfc2784>.

[RFC3484] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, DOI 10.17487/RFC3484, February 2003, <http://www.rfc-editor.org/info/rfc3484>.

[RFC3931] Lau, J., Ed., Townsley, M., Ed., and I. Goyret, Ed., "Layer Two Tunneling Protocol - Version 3 (L2TPv3)", RFC 3931, DOI 10.17487/RFC3931, March 2005, <http://www.rfc-editor.org/info/rfc3931>.

[RFC4213] Nordmark, E. and R. Gilligan, "Basic Transition Mechanisms for IPv6 Hosts and Routers", RFC 4213, DOI 10.17487/RFC4213, October 2005, <http://www.rfc-editor.org/info/rfc4213>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <http://www.rfc-editor.org/info/rfc4271>.

[RFC4374] McCobb, G., "The application/xv+xml Media Type", RFC 4374, DOI 10.17487/RFC4374, January 2006, <http://www.rfc-editor.org/info/rfc4374>.

[RFC6459] Korhonen, J., Ed., Soininen, J., Patil, B., Savolainen, T., Bajko, G., and K. Iisakkila, "IPv6 in 3rd Generation Partnership Project (3GPP) Evolved Packet System (EPS)", RFC 6459, DOI 10.17487/RFC6459, January 2012, <http://www.rfc-editor.org/info/rfc6459>.

[RFC7024] Jeng, H., Uttaro, J., Jalil, L., Decraene, B., Rekhter, Y., and R. Aggarwal, "Virtual Hub-and-Spoke in BGP/MPLS VPNs", RFC 7024, DOI 10.17487/RFC7024, October 2013, <http://www.rfc-editor.org/info/rfc7024>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <http://www.rfc-editor.org/info/rfc7432>.

13.2. Informative References

[I-D.drao-bgp-l3vpn-virtual-network-overlays]
Rao, D., Mullooly, J., and R. Fernando, "Layer-3 virtual network overlays based on BGP Layer-3 VPNs", draft-drao-bgp-l3vpn-virtual-network-overlays-03 (work in progress), July 2014.

[I-D.ietf-bess-evpn-overlay]
Sajassi, A., Drake, J., Bitar, N., Isaac, A., Uttaro, J., and W. Henderickx, "A Network Virtualization Overlay Solution using EVPN", draft-ietf-bess-evpn-overlay-01 (work in progress), February 2015.

[RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <http://www.rfc-editor.org/info/rfc4389>.

[RFC4761] Kompella, K., Ed. and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, DOI 10.17487/RFC4761, January 2007, <http://www.rfc-editor.org/info/rfc4761>.

[RFC7080] Sajassi, A., Salam, S., Bitar, N., and F. Balus, "Virtual Private LAN Service (VPLS) Interoperability with Provider Backbone Bridges", RFC 7080, DOI 10.17487/RFC7080, December 2013, <http://www.rfc-editor.org/info/rfc7080>.

[RFC7209] Sajassi, A., Aggarwal, R., Uttaro, J., Bitar, N., Henderickx, W., and A. Isaac, "Requirements for Ethernet VPN (EVPN)", RFC 7209, DOI 10.17487/RFC7209, May 2014, <http://www.rfc-editor.org/info/rfc7209>.

Authors' Addresses

Keyur Patel
Arrcus

Email: keyur@arrcus.com

Ali Sajassi
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95124 95134
USA

Email: sajassi@cisco.com

John E. Drake
Juniper Networks, Inc.

Email: jdrake@juniper.net

Zhaohui Zhang
Juniper Networks, Inc.

Email: zzhang@juniper.net

Wim Henderickx
Nokia

Email: wim.henderickx@nokia.com

BESS
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

W. Lin
Z. Zhang
J. Drake
Juniper Networks, Inc.
J. Rabadan
Nokia
A. Sajassi
Cisco Systems

March 13, 2017

EVPN Inter-subnet Multicast Forwarding
draft-lin-bess-evpn-irb-mcast-03

Abstract

This document describes inter-subnet multicast forwarding procedures for Ethernet VPNs (EVPN). This includes forwarding inside an EVN domain and to/from outside the EVPN domain.

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

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 http://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 September 14, 2017.

Table of Contents

1. Introduction 2
1.1. Background and Terminologies 3
1.1.1. Integrated Routing and Bridging 3
1.1.2. General Multicast Routing 4
1.2. Inter-subnet Multicast in EVPN 5
2. EVPN-aware Solution 7
2.1. Basic Operations 7
2.2. Multi-homing Support 8
2.3. Receiver NVEs not connected to a source subnet 9
2.3.1. IMET routes advertisement 10
2.3.2. Layer 2 Forwarding State 11
2.3.3. Layer 3 Forwarding State 12
2.4. Selective Multicast 12
2.5. Advanced Topics 14
2.5.1. Legacy NVEs 14
2.5.2. Traffic to/from outside of an EVPN domain 15
2.5.3. Integration with MVPN 17
2.5.4. When Tenant Routers Are Present 19
3. IANA Considerations 20
4. Security Considerations 21
5. Acknowledgements 21
6. References 21
6.1. Normative References 21
6.2. Informative References 21
Appendix A. Integrated Routing and Bridging 23
Authors' Addresses 24

1. Introduction

EVPN offers an efficient L2 VPN solution with all-active multi-homing support for intra-subnet connectivity over MPLS/IP network. EVPN also provides an integrated L2 and L3 service. When forwarding among Tenant Systems (TS) across different IP subnets is required, Integrated Routing and Bridging (IRB) can be used [ietf-bess-evpn-inter-subnet-forwarding].

An network virtualization endpoint (NVE) device supporting IRB is called a L3 Gateway. In a centralized approach, a centralized gateway provides all routing functionality, and even two tenant systems on two subnets connected to the same NVE need to go through the central gateway, which is inefficient. In a distributed approach, each NVE has IRB configured, and inter-subnet traffic will be locally routed without having to go through a central gateway.

Inter-subnet multicast forwarding is more complicated and not covered in [ietf-bess-evpn-inter-subnet-forwarding]. This document describes the procedures for inter-subnet multicast forwarding.

1.1. Background and Terminologies

For each Broadcast Domain (BD, an L2 concept), there is usually a subnet (an L3 concept). This document may use subnet and BD interchangeably. When inter-subnet forwarding is allowed between some subnets of the same tenant on the same NVE, the BDs are associated with the same routing instance via IRB interfaces. Multiple BDs of the same tenant may be attached to different routing instances if inter-subnet forwarding is subject to some restrictions. This document assumes that inter-subnet forwarding is allowed by default between subnets of the same tenant.

1.1.1. Integrated Routing and Bridging

Appendix A describes the concept of Integrated Routing and Bridging and in particular IRB interfaces in more details.

An IRB interface is a logical connection between a BD and a routing instance. It has two ends - one on routing instance side and one on BD side. In this document, when we say a packet is "routed/sent down an IRB interface", it is from L3 point of view and on the routing instance side (from L3 down to L2). L3 forwarding related processing like TTL/fragmentation and mac address change are done before the packet is put onto the IRB interface "wire" and sent to the corresponding BD. From the BD’s point of view, that packet is received on the BD side of the IRB interface and L2 switched out of one or more other L2 interfaces (Attachment Circuits or ACs) in the BD.

Note that there is one BD in a MAC-VRF with Vlan-based service and multiple BDs in a MAC-VRF with Vlan-aware Bundle Service. Therefore, a routing instance for a tenant may have one or more MAC-VRFs associated with it, with the IRB interfaces being the ties.

1.1.2. General Multicast Routing

IP routing is inter-subnet forwarding - traffic received from one subnet is routed/forwarded to other subnets. The subnets could be tradictional networks like LANs or could be broadcast domains implemented by EVPN. This section provides a very high level description on layer 3 multicast routing and is not specific to EVPN at all.

Multicast routing is based on trees - rooted at the source or Rendezvous Point (RP). Typically the tree is set up by PIM protocol [RFC7761] following the reverse path from a receiver towards the source/RP. On a particular router on the tree, the process to determine the upstream interface/neighbor is called the RPF process and the upstream interface/neighbor is also called the RPF interface/ neighbor. The PIM protocol signals the control plane state, and corresponding (s,g) or (*,g) forwarding state is installed on the routers on the tree. The forwarding state includes one (or more, in case of bidirectional trees) expected Incoming Interfaces (IIFs) and a list of Outging interfaces (OIFs). The IIF is the RPF interface (IIF is forwarding state while RPF is control plane state, but may be used interchangeably in this document) towards the source/RP, and in case of bidirectional trees [RFC5015], the IIFs also include other interfaces where traffic is accepted.

An interface is added to the OIF list if one of the following two conditions is met:

- There are local receivers on the subnet that the interface is connected to, and this router is the PIM Designated Router (DR) or IGMP/MLD Querier if PIM is not used. In this case the router is referred to as a Last Hop Router (LHR).

- A PIM join has been received from a downstream router connected by this interface.

The LHR also send PIM join messages towards its RPF neighbor. This will establish the branch of the tree towards the root.

In case of PIM-SM for ASM (Any Source Multicast), the LHRs send (*,g) joins towards the RP, establishing a (*,g) shared tree rooted at the RP. On the subnet that a source is connected to, the PIM DR, referred to as First Hop Router (FHR), sends PIM Register messages to the RP when it receives initial traffic for a flow. The RP then sends (s,g) PIM join towards the FHR, establishing a branch from the RP towards the source. Traffic is initially sent from the FHR to the RP following the (s,g) branch, and the RP delivers the traffic to all LHRs following the (*,g) shared tree. Upon receiving traffic, an LHR

optionally sends (s,g) join towards the source, establishing an (s,g) branch between the source and the LHR so that traffic can follow a more optimal path.

1.2. Inter-subnet Multicast in EVPN

For multicast traffic sourced from a TS in subnet 1, EVPN Broadcast, Unknow unicast, Multicast (BUM) forwarding based on RFC 7432, will deliver it to all sites in subnet 1. When NVEs receive the mulitcast traffic on IRBs for subnet1, they route the traffic to other subnets via their IRB interfaces following multicast routing procedures. From an L3 point of view, each NVE has an (IRB) interface to subnet 1, and hence is attached to the same subnet as the multicast source. Nothing is different from a traditional LAN and regular IGMP/MLD/PIM procedures kick in.

If a TS is a multicast receiver, it uses IGMP/MLD to signal its interest in some multicast flows. One of the gateways is the IGMP/MLD querier for a given subnet. It sends queries down the IRB for that subnet, which in turn causes the queries to be forwarded throughout the subnet following the EVPN BUM procedures. TS's send IGMP/MLD joins via multicast, which are also forwarded throughout the subnet via EVPN BUM procedure. The gateways receive the joins via their IRB interfaces. From layer 3 point of view, again it is nothing different from a traditional LAN.

On a traditional LAN, only one router can send multicast to local receivers on the LAN. That is either the PIM Designated Router (subject to PIM Assert procedure) or IGMP/MLD querier (if PIM is not used - e.g., the LAN is a stub network). On the source subnet, PIM is typically needed so that traffic can be delivered to other subnets via other routers. For example, in case of PIM-SM, the DR on the source network encapsulates the initial packets for a particular ASM flow in PIM Register messages and unicasts the Register messages to the Rendezvous Point (RP) for that flow, triggering necessary state for that flow to be built throughout the network.

That also works in the EVPN scenario, although not efficiently. Consider the example depicted in Figure 1, where a tenant has two subnets (subnets 1 and 2) corresponding to two EVPN broadcast domains (VLANs 1 and 2) at three sites. With VLAN-based service, each broadcast domain has its own EVI. With VLAN-aware bundle service, many broadcast domains can belong to the same EVI.

In Figure 1, a multicast source is located at site 1 on subnet 1 and three receivers are located at site 2 on subnet 1, site 1 and 2 on subnet 2 respectively. PIM adjacencies are formed among the NVEs on

each subnet. On subnet 1, NVE1 is the PIM DR while on subnet 2, NVE3 is the PIM DR.

Multicast traffic from the source at site 1 on subnet 1 is forwarded to all three sites on BD 1 following EVPN BUM procedure. Rcvr1 gets the traffic when NVE2 sends it out of its local Attachment Circuit (AC). The three gateways for EVI1 also receive the traffic on their IRB interfaces for subnet1 and potentially route to other subnets. NVE3 is the DR on subnet 2 so it routes the local traffic (from L3 point of view) to subnet 2 while NVE1/2 is not the DR on subnet 2 so they don't. Once traffic gets onto subnet 2, it is forwarded back to NVE1/2 and delivered to rcvr2/3 following the EVPN BUM procedures.

Notice that the traffic is sent across the EVPN core multiple times - once for each subnet with receivers. Additionally, both NVE1 and NVE2 receive the multicast traffic from subnet 1 on their IRB interfaces for subnet 1, but they do not route to subnet 2 where they are not the PIM DRs. Instead, they wait to receive traffic at L2 from NVE3. For example, for receiver 3 connected to NVE1 but on different IP subnet as the multicast source, the multicast traffic from source has to go from NVE1 to NVE3 and then back to NVE1 before it is being delivered to the receiver 3. This is similar to the hairpinning issue with centralized approach - the inter-subnet multicast forwarding is centralized via the DR, even though distributed approach is being used for unicast (in that each NVE is supporting IRB and routing inter-subnet unicast traffic locally).

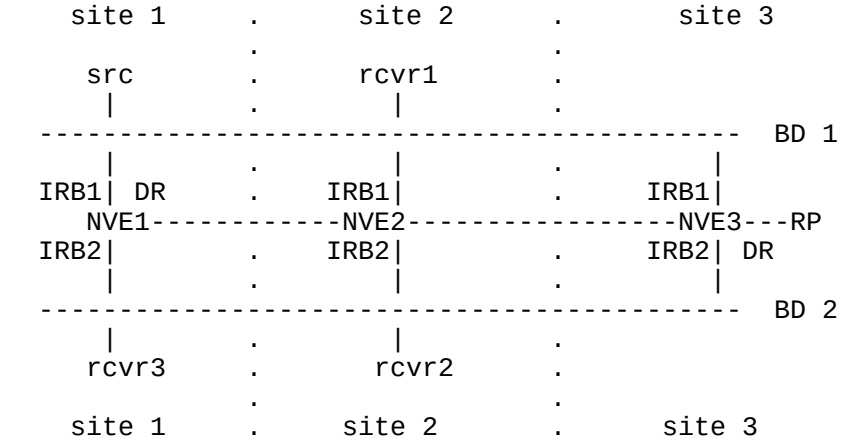


Figure 1 - EVPN IRB multicast scenario

2. EVPN-aware Solution

In the above text, the term "gateway" is from hosts point of view, referring to a "routing gateway" that provides layer 3 forwarding. With the distributed approach, each or almost every NVE is a gateway, hence in the rest of the document we simply use the term NVE instead of gateway.

2.1. Basic Operations

The multicast forwarding inefficiency described above (hairpinning and multiple copies across the core) can be avoided if the following Optimized Inter-subnet Multicast (OISM) procedures are followed:

1. When a routing instance on an NVE receives multicast traffic on one of its IRB interfaces, it routes the traffic down any other IRB interfaces that attach to subnets that have receivers for the traffic, regardless whether the NVE is DR for those IRB interfaces or not.

2. For ASM multicast traffic sourced from a local AC, if PIM runs on the corresponding IRB interface, the NVE behaves as if it were the DR on the IRB interface and performs PIM Registering procedures.

3. When an NVE receives Membership Reports from one of its ACs and PIM runs on the corresponding IRB interface, it sends PIM joins towards the RP or source regardless if it is DR/querier or not.

4. Multicast data traffic received by a BD on its IRB interface (i.e. multicast data traffic routed down the IRB interface) is L2 switched out of that BD’s local ACs only and not forwarded to other NVEs. Note that link local multicast traffic (e.g. addressed to 224.0.0.x in case of IPv4), is not subject to the above procedures. It is still forwarded to remote NVEs in the same subnet following EVPN procedures and not routed into other subnets.

The above procedures are for routing traffic from the source subnet to other subnets. In the source subnet itself, traffic is L2 switched according to EVPN procedures. It is assumed that each NVE of the tenant can receive the L2 switched traffic in the source subnet. If there are NVEs not attached to every subnet (therefore an NVE cannot received L2 switched traffic in a source subnet that it is not connected to), then a Supplemental BD (Section 2.3) is needed to L2 switch the traffic from the source NVE to NVEs not attached to the source subnet. In that SBD, multicast data traffic received on its IRB interface is forwarded to other NVEs, as an exception to rule 4.

That is needed for situations discussed in Section 2.5.2 and Section 2.5.4.

In the example in Figure 1, when NVE1's routing instance receives traffic on its IRB1 interface it will route the traffic down its IRB2 for delivery to local rcvr3. It also sends register messages to the RP since the source is local. Both NVE2 and NVE3 will receive the traffic on IRB1 but neither sends register messages to the RP, since the source is not local. NVE2 will route the traffic down its IRB2 and deliver to local rcvr2. NVE3 will also route the traffic down IRB2 even though there is no receiver at the local site, because the IGMP/MLD joins from rcvr2/3 are also received by NVE3.

Essentially, each NVE behaves as a DR/querier on an IRB interface for local senders and receivers, and multicast data traffic routed down IRB interfaces is limited to local receivers.

If EVPN is only used to provide DC overlay service but not transit service (i.e. simulate a transit LAN connecting tenant routers) for a tenant, then there is no need to run PIM protocol and the rule 2 and 3 above do not apply. Otherwise, additional procedures in Section 2.5.4 are needed.

2.2. Multi-homing Support

The solution works as described when there are multi-homed ethernet segments.

As shown in Figure 2, both rcvr4 and rcvr5 are all-active multi-homed to NVE2 and NVE3. Receiver 4 is on subnet BD 1 and receiver 5 is on BD 2. When IRBs on NVE1 and NVE2 forward multicast traffic to its local attached access interface(s) based on EVPN BUM procedure, only DF for the ES deliveries multicast traffic to its multi-homed receiver. Hence no duplicated multicast traffic will be forwarded to receiver 4 or receiver 5.

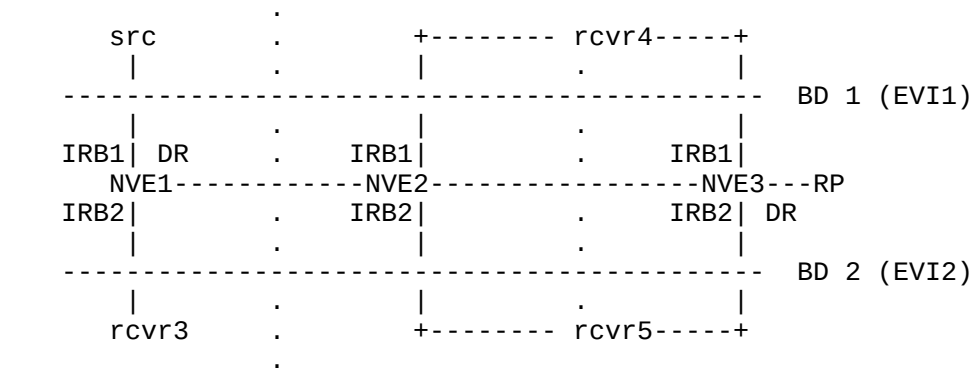


Figure 2 - EVPN IRB multicast and multi-homing

For traffic sourced from a multi-homed ES, existing split-horizon procedures work as is, because vanilla EVPN forwarding is used for intra-subnet traffic.

2.3. Receiver NVEs not connected to a source subnet

The procedures of this document require that a inter-subnet multicast packet is carried across the core as an intra-subnet frame. However, consider the case where, for a given tenant, (a) NVE-1 attaches to subnet-1, (b) NVE-2 attaches to subnet-2 but not to subnet-1, and (c) a receiver in subnet-2 needs to receive multicast packets that are sourced in subnet-1. Since NVE-1 sends the packets across the core as intra-subnet multicasts, how does NVE-2 receive the packets?

One possible solution would be to configure subnet-1 on NVE-2. On NVE-2, subnet-1 would have an IRB interface attaching it to the routing instance, but subnet-1 would have no ACs. Then NVE-2 would receive the intra-subnet multicast traffic of subnet-1, and the procedures already discussed would cause the traffic to be forwarded to NVE-2’s local ACs for subnet-2.

However, if a given tenant has many subnets, only a few of which attach to any given NVE, it is undesirable to have to configure all those subnets on all those PEs. To avoid this, we introduce the notion of a "Supplemental Broadcast Domain" (SBD). Each NVE will have a single SBD (per tenant) configured. The SBD has no ACs, just an IRB interface. The purpose of the SBD on a given NVE is to receive (over the core) the intra-subnet multicasts of all subnets that are not atached to that NVE. Additionally, traffic routed down the SBD IRB interface will be sent across the core to remote NVEs. This is an exception to rule 4 in Section 2, and is explained in Section 2.5.2 and Section 2.5.4.

Thus in the above example, when NVE-1 sends a multicast packet from subnet-1 to other NVEs, NVE-2 will receive the packet on the SBD. Note that, in the example, NVE-1 would not have to send any extra copies of the packet across the core. It just sends what it would normally send. If an NVE receiving the packet is attached to subnet-1, it associates the packet with subnet-1; if an NVE receiving the packet is not attached to subnet-1, it associates the packet with the SBD.

Subsequent sections explain how the NVEs construct the necessary EVPN routes to make this happen.

2.3.1. IMET routes advertisement

The SBD is a separate broadcast domain present on all the NVEs of the tenant. It has a corresponding IRB interface but no ACs. With VLAN-based service, the SBD is in its own EVI. With VLAN-aware bundle service, the SBD is just an additional BD in the EVI. The SBD uses a Route Target that allows its routes to be imported by all the NVEs of the tenant and associated with the SBD. In case of VLAN-aware bundle service, the Route Target may be the same as or different from the Route Targets for other BDs in the same EVI. In this document, when we say a route is originated for/in the SBD, it means that the RD of the route is set to the RD of the originating NVE's MAC-VRF for the SBD, the Route Target is set to that of the SBD, and the Tag ID is set to 0 in case of VLAN-based service or the Tag ID for the SBD in case of VLAN-aware bundle service.

The rules of IMET route advertisement can be summarized as following:

- When IR, BIER, or RSVP-TE P2MP is being used for inclusive tunnels, each NVE originates an IMET route in the SBD. In case of IR, the MPLS Label field in the IMET route's PMSI Tunnel Attribute (PTA) is a downstream allocated label for the SBD.

- When PIM, BIER or mLDP/RSVP-TE P2MP is being used for inclusive tunnels, the IMET route that an NVE originates for a subnet carries the RT for the subnet and the RT for the SBD.

- In case of BIER, or if tunnel aggregation (a single tunnel is used for more than one broadcast domains) is used for mLDP/RSVP-TE P2MP, the IMET route for the source subnet carries an upstream allocated label in the PMSI Tunnel Attribute. The label is different for each source subnet.

With the above rules, IMET routes are advertised in both the SBD and source subnets if IR, BIER or RSVP-TE P2MP tunnels are used. IMET

routes are only advertised in the source subnet in case of PIM/mLDP P2MP tunnels.

2.3.2. Layer 2 Forwarding State

In case of IR, when a source NVE builds its L2 forwarding state for a BD, it finds all the remote NVEs that needs to receive traffic by finding the IMET routes for the SBD. The IMET routes for the SBD are those in the MAC-VRF for the SBD (in case of VLAN-based service) or those in the MAC-VRF for the SBD and with the SBD's Tag ID (in case of VLAN-aware bundle service).

If a remote NVE (learnt via the IMET route for the SBD) also advertises an IMET route for the source subnet, the label in that route is used. Otherwise, the label in the IMET route for the SBD is used. Thus when a packet is transmitted to an NVE attached to the source subnet, it carries the label that that NVE assigned to the source subnet. When a packet is transmitted to an NVE that is not attached to the source subnet, it carries the label that that NVE assigned to the SBD.

In case of RSVP-TE P2MP, the source NVE establishes a P2MP tunnel to all remote NVEs found through the SBD's IMET routes and advertises the tunnel in the IMET route for the source subnet. If tunnel aggregation is not used, a remote NVE attached to the source subnet binds the incoming tunnel branch to the source subnet, and a remote NVE that is not attached to the source subnet binds the incoming tunnel branch to the SBD.

In case of PIM/mLDP, a remote NVE joins the tunnel advertised in the IMET route for a source subnet. If tunnel aggregation is not used, a remote NVE attached to the source subnet binds the incoming tunnel branch to the source subnet, and a remote NVE that is not attached to the source subnet binds the incoming tunnel branch to the SBD.

In case of BIER, or if tunnel aggregation is used for mLDP/RSVP-TE P2MP, a remote NVE binds the upstream allocated label in the IMET route for a source subnet to that subnet if it is present on the NVE. Otherwise it binds the label to the SBD.

With the forwarding state set up as above, the incoming traffic from a remote NVE is either associated with the source subnet or with the SBD. In the former case, traffic is forwarded at L2 to local receivers in the same source subnet, and split-horizon procedures for multi-homing work as is. In the latter case, the traffic appears to the receiving NVE as if it were sourced from the SBD.

The incoming traffic from a remote NVE is also associated with the IRB interface in either the source subnet or SBD and routed down other IRB interfaces for local receivers in other subnets, according to a matching Layer 3 forwarding state described in the following section.

2.3.3. Layer 3 Forwarding State

When an NVE's routing instance receives IGMP/MLD joins on IRB interfaces, corresponding (C-S,C-G) or (C-*,C-G) L3 forwarding entries are created/updated. The OIF list includes IRB interfaces that have corresponding (C-S,C-G) or (C-*,C-G) IGMP/MLD state built from relevant IGMP/MLD joins. An OIF is removed when the corresponding IGMP/MLD state is removed from the interface, and the (C-S,C-G) or (C-*,C-G) L3 forwarding state is removed when all of its OIFs are removed.

For (C-S,C-G) L3 forwarding entries, the IIF is set to the source subnet's IRB interface if the source subnet is present on the NVE. If the source subnet is not present on the NVE, the IIF is set to the SBD's IRB interface.

For (C-*,C-G) forwarding entries, the RPF interfaces include all IRB interfaces as the traffic can arrive in the SBD or in any subnet to which the NVE is attached. Note that for a particular packet, it only arrive once, and is associated with either the source subnet or the SBD.

2.4. Selective Multicast

For intra-subnet selective multicast, [I-D.sajassi-bess-evpn-igmp-mld-proxy] specifies the procedures of SMET routes. If a NVE has local receivers for (C-*,C-G) traffic in subnet X, since the sources could be in any of other subnets that are present on the NVE, it would need to advertise the (C-*,C-G) SMET routes in each of those source subnets to pull traffic. To avoid the duplication, SBD is used even if every subnet is connected to every NVE of a tenant, and SMET routes are advertised as following:

- If there are tenant routers (Section 2.5.4), SMET routes are originated per [I-D.sajassi-bess-evpn-igmp-mld-proxy] in the subnet where the state is originally learnt. This will allow NVEs in the same subnet to convert SMET routes back to IGMP/MLD messages on ACs.

- Additionally, a corresponding SMET route is originated for the SBD, with the v1/v2/v3 flag bits cleared, with one exception described below.

Note that for (C-S,C-G) SMET routes, even though they would not need to be advertised in every source subnet like in (C-*,C-G) case, they are also advertised in the SBD. The reason is that a receiver for an (C-S,C-G) flow may be attached to a NVE that is not connected to the source subnet so the SMET route need to be advertised in the SBD anyway in that case. For consistence in all situations, all SMET routes are advertised in the SBD.

The one exception is that a (C-S,C-G) SMET route with the IE (include/exclude) bit set may be suppressed in the SBD, according to the IGMP/MLD state merged from all subnets. For example, a particular source may be excluded in one subnet but not in another, then the SMET route will not be originated for the SBD. This can be considered that IGMP/MLD state in subnets is proxied into the SBD, just like the IGMP/MLD state on ACs is proxied to other ACs in the same subnet.

The SMET routes in the SBD will trigger IGMP/MLD state on the SBD's IRB interfaces. Note that for L3 multicast forwarding state, the SBD IRB interface is not added to the Outgoing InterFace (OIF) List when the RPF interface is one or more IRB interfaces (i.e., traffic is sourced from a BD), even with the IGMP/MLD state on the SBD IRB interface. The reason is that traffic from that BD is already L2 switched to all NVEs.

[I-D.sajassi-bess-evpn-igmp-mld-proxy] assumes selective forwarding is always used with IR or BIER for all flows. The SMET route allows other NVEs to identify which NVEs need to receive traffic for a particular (C-S,C-G) or (C-*,C-G). With SBD, a source NVE builds the corresponding forwarding state using the same procedure as in the inclusive tunnel case, except that it checks the corresponding SMET route in the SBD to determine if a remote NVE needs to receive the traffic.

For other tunnel types, or if selective forwarding is only used for some of the flows, S-PMSI A-D routes are needed as specified in [I-D.ietf-bess-evpn-bum-procedure-updates]. A source NVE advertises S-SPMSI A-D routes to announce the tunnels used for certain flows, and receiving NVEs either join the announced PIM/mLDP tunnel or respond with Leaf A-D routes if the Leaf Information Requested flag is set in the S-PMSI A-D route's PTA (so that the source NVE can include them as tunnel leaves). As in the inclusive tunnel case, the S-PMSI A-D routes additionally carry the RT for the SBD so that all NVEs of the tenant will import them. A receiving NVE binds the announced tunnel to either the subnet that the route is for if the subnet is present on the NVE or to the SBD otherwise.

2.5. Advanced Topics

2.5.1. Legacy NVEs

It is possible that an NVE may not support the OISM procedures. For example, it may not have IRB interfaces for some of its BDs, or its software could not be upgraded to support OISM. To indicate the OISM support, an NVE that supports the procedures in this document includes the Multicast Flags Extended Community in its IMET routes and sets a new flag bit (OISM bit, to be assigned by IANA) in the EC.

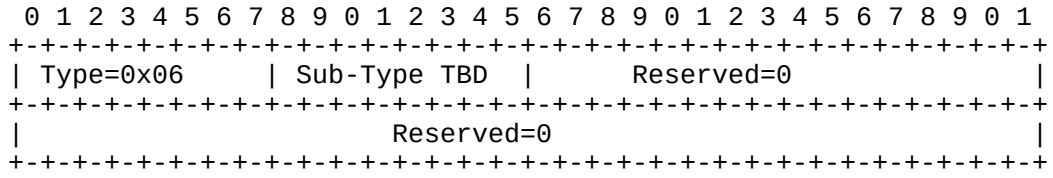
Suppose a multicast source is attached to NVE 1 in subnet 1. Subnet 1 is not present on NVE 2 that does not support OISM, and NVE 2 has some receivers in its subnet 2. In this case, the receivers need to receive traffic in subnet 2 from NVE 1. For that, the OISM NVEs run PIM over the subnet for which not all NVEs support OISM, and the elected PIM DR use a separate provider tunnel to forward traffic (that is routed down the DR's IRB interface for the subnet) only to NVEs that do not support OISM.

If the PIM DR uses IR to forward BUM traffic in the subnet, the special tunnel's leaves includes the NVEs that do not set the OISM bit in the above mentioned EC.

If the PIM DR uses P2MP tunnels, the special tunnel is advertised in an EVPN S-PMSI A-D route per [I-D.ietf-bess-evpn-bum-procedure-updates]. The route carries an EVPN Non-OISM Extended Community, indicating that a receiving NVE attached to the BD identified in the route should join the advertised tunnel only if it does not support OISM.

The routes could be either be a (C-*,C-*) wildcard S-PMSI A-D routes if an inclusive tunnel is used (but only for all sites without IRBs), or individual (C-S,C-G)/(C-*,C-G)/(C-S,C-*) S-PMSI A-D routes if selective tunnels are used. They are advertised for each of BD to deliver multicast traffic routed down the IRB interface for the BD to remote sites that do not have IRBs for the BD. If the same (C-S,C-G)/(C-*,C-G)/(C-S,C-*)/(C-*,C-*) S-PMSI A-D routes are also advertised without the EVPN Non-OISM EC (to deliver intra-subnet traffic), then different RDs MUST be used for the two routes.

The EVPN Non-OISM Extended Community is a new EVPN extended community. EVPN extended communities are transitive extended community with a Type field of 6. The subtype of this new EVPN extended community will be assigned by IANA, and with the following 8-octet encoding:



For multicast sources attached to a Non-OISM NVE, if the source subnet is present on all NVEs, then traffic will be L2 switched to all NVEs in the source subnet and then forwarded appropriately. For simplicity, this document requires that all subnets on a Non-OISM NVE are configured on all NVEs, even if there would be no ACs on some NVEs for those subnets.

2.5.2. Traffic to/from outside of an EVPN domain

For traffic coming in/out of an EVPN domain, EVPN Gateways (GWs) are used. They are NVEs that also participate in the SBD for each tenant, and may be connected to some subnets. This document supposes that the GWs run PIM on its external tenant interfaces, or act as MVPN PEs for external connection (and the IRB interfaces are VRF interfaces in the IPVPN). The subnets in the EVPN domain appear as stub networks connected to the PIM/MVPN domain. This section describes the procedures that are common for both PIM and MVPN as external connection, while the next section focuses on procedures specific to MVPN.

If there are multiple GWs for the same EVPN domain, then the GWs need to run PIM on the IRB interfaces for the subnets and the SBD, so that a DR can be elected for each subnet/SBD, and act as FHR/LHR on the subnets/SBD. In other words, traffic inside the EVPN domain follows the procedures described in previous sections, while traffic to/from outside the EVPN domain need to additioanlly follow existing PIM/MVPN procedures.

For traffic going out of the EVPN domain, the IRB interface of the source subnet or SBD is the RPF interface on the GW, depending on whether the source subnet is present on the GW. In case of PIM-SM, one of the EVPN GWs is the PIM DR on a connected source subnet or on the SBD act as the First Hop Router (e.g. handling PIM register procedures for ASM). For that, the SBD IRB needs to be configured to treat incoming packets as if the sources were on a local subnet (in this case the SBD).

When selective forwarding is used in the EVPN domain, for the EVPN GW to receive all traffic (before it learns possible external receivers) for the purpose of FHR procedures, it MUST advertise a (C-*,C-*) SMET

route in the SBD, indicating to other NVEs that it needs to receive all traffic. Later the EVPN GW may receive (C-S,C-G) prunes from the external network. At that time, it MAY advertise (C-S,C-G) SMET route with the Exclude Group type bit and IGMPv3 bit in the Flags field set, signaling to other NVEs that the particular (C-S,C-G) traffic is not needed.

For traffic coming into a EVPN domain, the IRB interfaces for connected subnets are included in OIF list for the L3 multicast forwarding route, if the subnets have corresponding local IGMP/MLD state. The IRB interface of the SBD may also be added as an outgoing interface so that remote NVEs can receive the traffic and route to their connected subnets. Note that in this case, data traffic sent down the SBD IRB interface is forwarded to remote NVEs (this is a exception to the behavior in Section 2). The SBD IRB interface is added only if the GW has corresponding SMET routes (as described in Section 2.4) received from other NVEs in the SBD. Corresponding PIM join/prune messages or BGP-MVPN routes will be triggered/withdrawn as a result.

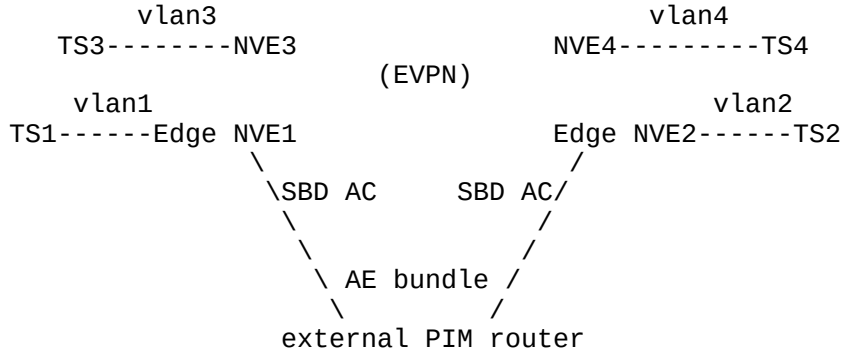
For (C-*,C-G) L3 forwarding state, Section 2.3.3 states that all IRB interfaces are included in the RPF interface list. Section 2.4 states that the SBD IRB interface is not added to OIF list if the RPF interfaces include one or more IRB interfaces. That is to prevent routing internal traffic into the SBD at layer 3 (because the source NVE already L2 switch the traffic to all NVEs). This means that traffic coming into the EVPN domain cannot use the (C-*,C-G) forwarding state (it would not be routed down the IRB interface for ths SBD to reach remote NVEs because that IRB is not in the OIF list). For this to work, the interface or MVPN tunnel connecting towards the C-RP is not added as an IIF of the (C-*,C-G) forwarding state (even though a PIM join is sent out of that interface), so initial traffic for an externally sourced flow will match the (C-*,C-G) forwarding state and trigger IIF Mismatch notifications, (since the incoming interface does not match any of the IIFs), causing the EVPN GW to install (C-S,C-G) state with the external interface (or MVPN provider tunnel) being the RPF interface and IRB interface included in the OIF list.

2.5.2.1. A Variation of External Connection

If a tenant’s external connection can be via a vlan (instead of MVPN), and there are no sources like C-S1/2/5 as described in Section 2.5.3, then the following variation can be used.

The external vlan connection becomes an AC in the SBD. The tenant external router becomes the PIM FHR and LHR for the EVPN domain that is treated as a stub network. The previous EVPN GWs are no longer

gateways and are referred to as edge NVEs in this section. An AE bundle can be used to connect to multiple edge NVEs - the bundle terminates either on the external router or on a switch between the edge NVEs and the external router, as depicted in the following picture. From the edge NVEs' point of view, the external PIM router is a TS on a multihomed ES.

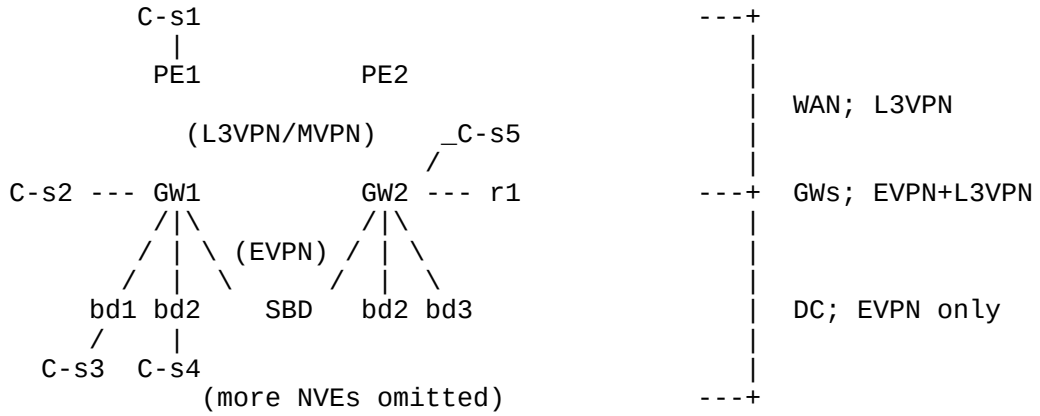


PIM is not running on any of the NVEs. IGMP/MLD state inside the EVPN domain is proxied to the external vlan and triggers corresponding multicast state on the external router. Externally sourced traffic is routed to the vlan as a result, and is L2 switched by the edge NVEs to other NVEs via the SBD. All receivers in the EVPN domain receive the traffic that is routed to them by their attached NVEs (the IIF is the SBD IRB and the OIFs are the IRBs for the subnets that the receivers are on).

For traffic sourced from inside the EVPN domain to reach external receivers, the edge NVEs still need to advertise a (C-*,C-*) SMET route in the SBD to pull all traffic and L2 switch to the external router, who will register towards the RP. The external router may prune back a particular flow by sending approproate IGMP/MLD messages, triggering correspoding SMET routes on the edge NVEs so that the source NVEs will stop sending traffic towards the edge NVEs.

2.5.3. Integration with MVPN

When a tenant needs to connect its EVPN subnets to external networks via L3VPN, instead of running both EVPN and L3VPN on each NVE, this document recommends that L3VPN (hence MVPN) only extends to the EVPN GWs, and only EVPN runs inside the EVPN domain. EVPN GWs run both EVPN and L3VPN/MVPN, as depicted in the following diagram.



GW1/2 run both EVPN and L3VPN. They may advertise routes learnt from PE1/PE2 (e.g. C-s1), routes to locally attached non-EVPN destinations (e.g., C-s2/s5), or just a default route into the EVPN domain as EVPN type-5 routes. For destinations inside the EVPN domain (including EVPN and non-EVPN, e.g. C-s2/3/4/5), the GWs may advertise subnet prefix L3VPN routes towards outside the EVPN domain, or optionally advertise host IPVPN route when they’re learnt via EVPN type-2 routes. The L3VNP routes are all advertised with Source AS and VRF Route Import ECs [RFC6514] for MVPN purpose.

Using the GW2 example, when it determines RPF interface/neighbor or MVPN UMH for various sources, it follows the following rules:

- If the source (e.g. C-s5) is reachable on a local non-IRB interface, use that interface as the RPF interface. Or,
- If the source (e.g. C-s4) is on a local BD, use the IRB for that local subnet as the RPF interface. Or,
- If the route to the source (e.g. C-s2/s3) is learnt via EVPN type-2/5 routes, use the SBD IRB as the RPF interface. Or,
- If the route to the source (e.g. C-s1/s2) has a VRF Import RT EC, then use MVPN procedure for UMH selection and use the MVPN provider tunnel as the RPF interface.

Notice that for C-s2, GW2 may either use the SBD IRB or the MVPN provider tunnel as the RPF interface, depending whether the IPVPN route or EVPN type-5 route is selected as the active route.

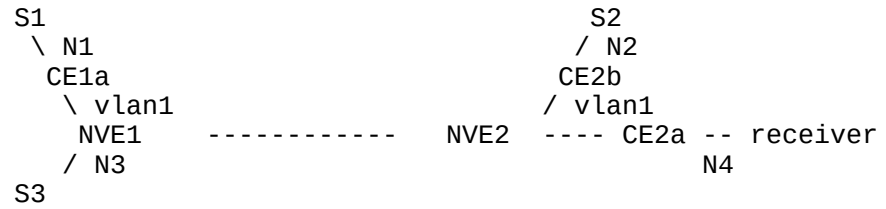
Also notice that for C-s4, if GW1/2 only advertises the subnet prefix into L3VPN, then PE1/2 may pick GW2 as the UMH. It will still work as GW2 will get the traffic in bd2 as well. However, it would be

more optimal if GW1 is picked as the UMH as C-s4 is directly attached to GW1. To achieve this optimization, when GW2 receives the C-multicast route for (C-s4,C-g) from PE1/2, it may optionally advertise a C-multicast route to GW1 where C-s4 is directly attached. This will trigger an (C-s4,C-g) Source Active route, which PE1/2 may optionally use to influence their UMH selection such that GW1 is chosen as their UMH for C-s4.

2.5.4. When Tenant Routers Are Present

It is possible that an EVPN broadcast domain is providing transit service for a tenant's larger network and there are tenant routers attached to the subnet, running routing protocols like PIM. In that case, traffic routed by an upstream NVE to the subnet via IRB interface may be expected on a downstream tenant router. However, since multicast data traffic sent down the IRB interfaces is forwarded to local ACs only and not to other EVPN sites according to rule 4 in Section 2, additional procedures are needed to handle this situation with tenant routers. In particular, NVEs connecting to tenant routers or traffic sources need to run PIM on the IRB interface for the transit subnet and the SBD.

Consider the following situation:



CE1a, CE2a/b are three CE routers on vlan1 that is implemented by EVPN. The CEs and NVE1/2 run PIM protocol and are PIM neighbors on vlan1. CE2a has a receiver on network N4 for multicast traffic from S1/2/3 on network N1/2/3 respectively.

CE2a sends PIM joins to CE1a/CE2b/NVE1 on vlan1 for the three sources respectively and they all route traffic accordingly onto vlan1. Traffic from S1/2 will reach CE2a because NVE1/2 receive the L2 traffic on their ACs and forward across the core following EVPN procedures. Traffic from S3 is routed into vlan1 by NVE1 via the IRB interface, and per rule 4 in Section 2 the traffic will not be sent across the core. Thus, according to the procedures specified so far, the traffic from S3 will never be received by NVE2 or CE2a.

To solve this problem, NVE2 needs to know that CE2a sent a PIM join to another NVE in vlan1 and needs to pull traffic via the SBD, where

the traffic via IRB is not blocked on the core side. Because PIM protocol already requires a router to process join/prune messages that it receives on an interface even if it is not the intended RPF neighbor (for the purpose of join suppression and prune overriding), NVE2 can realize that the upstream router in the join message is another NVE vs. a CE router (this only requires the NVEs to keep track if a neighbor is an NVE for the subnet). In that case, it treats that join/prune as for itself. Correspondingly, its PIM upstream state machine will choose one of the NVEs as the RPF neighbor. Between this local NVE and the chosen RPF neighbor there could be multiple subnets including the SBD but the SBD IRB interface is explicitly chosen as the RPF interface. Corresponding join/prune is sent over the SBD IRB interface (optionally the the join/prune could be replaced with SMET routes) and the upstream NVE will route traffic through the SBD. This NVE then route traffic further downstream to CE routers.

Similarly, if an NVE needs to send PIM join/prune messages due to its local IGMP/MLD state changes, the RPF interface is always explicitly set to the SBD IRB.

Note that, if CE2a chooses NVE1 or NVE2 instead of CE1a as its RPF neighbor for S1, then both CE1a and NVE2 will send traffic to vlan1 (NVE1 receives join from NVE2 on the SBD and sends join to CE1a on vlan1. NVE1 receives traffic from CE1a on vlan1 and route to SBD. NVE2 receives traffic on SBD and route to local receivers on vlan1). PIM assert procedure kicks in but only on NVE2, as CE1a does not receive traffic from NVE2. To address this, an NVE must track all the RPF neighbors and not add an IRB interface to the OIF list if it received a corresponding PIM join on the IRB, in which a tenant router is listed as the upstream neighbor. That tenant router will deliver traffic to the subnet, and the traffic will be forwarded through the core as it is not routed down the IRB but received on an AC.

With PIM-ASM, if the DR on a source subnet is a tenant router, it will handle the registering procedures for PIM-ASM. As a result, the NVE at same site as the tenant router/DR MUST not handle registering procedures as described in Section 2.

3. IANA Considerations

This document requests the following IANA assignments:

- A "Non-OISM" Sub-Type in "EVPN Extended Community Sub-Types" registry for the EVPN Non-OISM Extended Community.

- An "Optimized Inter-subnet Multicast" bit (OISM) in the Multicast Flags extended community defined in [I-D.sajassi-bess-evpn-igmp-mld-proxy].

4. Security Considerations

To be updated.

5. Acknowledgements

The authors thanks Eric Rosen for his detailed review, valuable comments/suggestions and some suggesgted text. The authors also thanks Vikram Nagarajan and Princy Elizabeth for their contribution of the external connection variation (xref target="variation"/>. The authors alse benefited tremendously from the discussions with Aldrin Isaac on EVPN multicast optimizations.

6. References

6.1. Normative References

[I-D.ietf-bess-evpn-bum-procedure-updates]
Zhang, Z., Lin, W., Rabadan, J., and K. Patel, "Updates on EVPN BUM Procedures", draft-ietf-bess-evpn-bum-procedure-updates-01 (work in progress), December 2016.

[I-D.sajassi-bess-evpn-igmp-mld-proxy]
Sajassi, A., Thoria, S., Patel, K., Yeung, D., Drake, J., and W. Lin, "IGMP and MLD Proxy for EVPN", draft-sajassi-bess-evpn-igmp-mld-proxy-01 (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>.

[RFC7432] Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <http://www.rfc-editor.org/info/rfc7432>.

6.2. Informative References

[I-D.ietf-bess-evpn-inter-subnet-forwarding]
Sajassi, A., Salam, S., Thoria, S., Drake, J., Rabadan, J., and L. Yong, "Integrated Routing and Bridging in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-03 (work in progress), February 2017.

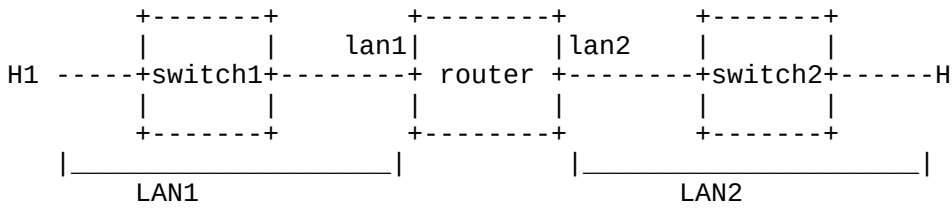
[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, DOI 10.17487/RFC4364, February 2006, <http://www.rfc-editor.org/info/rfc4364>.

[RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <http://www.rfc-editor.org/info/rfc5015>.

[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, <http://www.rfc-editor.org/info/rfc7761>.

Appendix A. Integrated Routing and Bridging

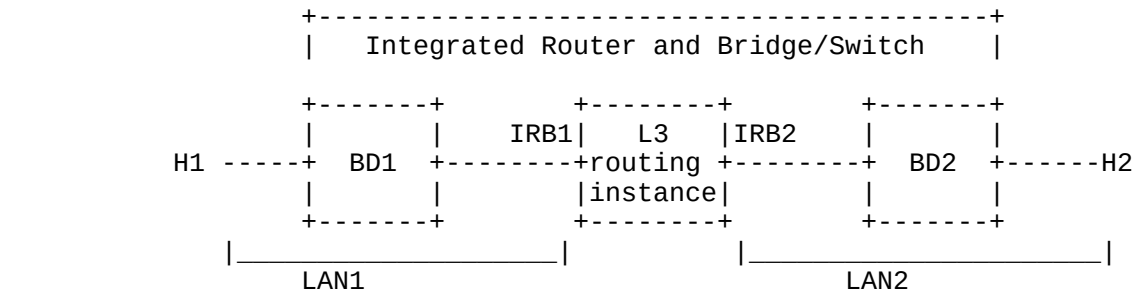
Consider a traditional router that only does routing and has no L2 switching (also referred to as "bridging") capabilities. It has two interfaces lan1 and lan2 connecting to LAN 1 and LAN 2 respectively. The two LANs are realized by two switches respectively, with hosts and the router attached:



Interfaces lan1 and lan2 are two physical interfaces with IP configuration and functionality. With that they may also be referred to as IP interfaces (on top of layer 2 interfaces). H1 has a default gateway configured, which is the router's IP address on interface lan1. For H1 to send an IP packet destined to H2, it uses the router's mac address (learnt via ARP resolution for the gateway) for lan1 as the destination mac address. The router receives the packet from the switch and associate it with the IP interface lan1 because the desitnation mac address matches. A IP lookup is done and the packet is sent out of interface lan2, with H2's mac address (again learnt via ARP resolution) as the destination mac address and the router's mac address on lan2 as the source mac address. TTL is decremented and fragmentation may be done during this forwarding process. This process may be referred to as "routing a packet". For comparison, when switch1 sends the packet that it receives from H1 to the router, it is "bridging or L2 switching a packet". There is no TTL or fragmentation for L2 switching, and there is no source/ destination mac address change.

If H1 sends an IP multicast packet, the multicast destination mac address and IPv4/6 Ethertype cause the router to associate the packet with the IP interface lan1 and may route it out of other IP interfaces as appropriate, following multicast routing rules.

Now consider that the router itself supports both routing and bridging. Now the above picture becomes the following:



The router now includes a routing instance (which could be the default/master instance, a Virtual Router routing instance, a VRF, or a VRF Lite, depending on which vendor's nomenclature one is familiar with) and two broadcast domains (BDs) that provide bridging functionalities. Instead of two physical interface connecting to two physical switches, there are two logical interfaces connecting the routing instance to the two BDs.

Because the device now provides both routing and bridging functionalities, it becomes an Integrated Router and Bridge(/Switch), and the two logical interfaces are referred to IRB interfaces, or sometimes simply IRBs. For each BD that needs routing functionality, there is one IRB interface connecting the BD to a particular routing instance.

Other than that the logical IRB interfaces replace physical interfaces, the way the packets are forwarded does not change.

Authors' Addresses

Wen Lin
Juniper Networks, Inc.

EMail: wlin@juniper.net

Zhaohui Zhang
Juniper Networks, Inc.

EMail: zzhang@juniper.net

John Drake
Juniper Networks, Inc.

EMail: jdrake@juniper.net

Jorge Rabadan
Nokia

EMail: jorge.rabadan@nokia.com

Ali Sajassi
Cisco Systems

EMail: sajassi@cisco.com

BESS Working Group
Internet Draft
Intended status: Standards Track
Expires: September 7, 2017

Y. Liu
F. Guo
Huawei Technologies
X. Liu
Jabil
R. Kebler
Juniper Networks
M. Sivakumar
Cisco
March 7, 2017

Yang Data Model for Multicast in MPLS/BGP IP VPNs
draft-liu-bess-mvpn-yang-03

Status of this Memo

Copyright Notice

Abstract

This document defines a YANG data model that can be used to configure and manage multicast in MPLS/BGP IP VPNs.

Table of Contents

1. Introduction 2
   1.1. Requirements Language..... 3
   1.2. Terminology 3
2. Design of Data model..... 3
   2.1. Scope of model 3
   2.2. Optional capabilities..... 3
   2.3. Position of address family in hierarchy..... 4
3. Module Structure 4
   3.1. MVPN Configuration..... 4
   3.2. MVPN Operational State..... 7
4. MVPN YANG Modules 12
5. Security Considerations..... 30
6. IANA Considerations 30
7. References 30
   7.1. Normative References..... 30
   7.2. Informative References..... 31
8. Acknowledgments 31

1. Introduction

YANG[RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF[RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces(e.g. REST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and Programmatic APIs.

This document defines a YANG data model that can be used to configure and manage Multicast in MPLS/BGP IP VPN(MVPN). It includes Cisco systems' solution [RFC6037], BGP MVPN [RFC6513] [RFC6514] etc. Currently this model is incomplete, but it will support the core MVPN protocols, as well as many other features mentioned in separate MVPN RFCs. In addition, Non-core features described in MVPN standards other than mentioned above RFC in future version.

1.1. 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 RFC-2119 [RFC2119].

1.2. Terminology

The terminology for describing YANG data models is found in[RFC6020].

This draft employs YANG tree diagrams, which are explained in [I-D.ietf-netmod-rfc6087bis].

2. Design of Data model

2.1. Scope of model

The model covers Rosen MVPN [RFC6037], BGP MVPN [RFC6513] [RFC6514]. The representation of some of extension features is not completely specified in this draft of the data model. This model is being circulated in its current form for early oversight and review of the basic hierarchy.

The operational state fields of this model are also incomplete, though the structure of what has been written may be taken as representative of the structure of the model when complete.

This model does not cover other MVPN related protocols such as MVPN Extranet [RFC7900] or MVPN MLDP In-band signaling [RFC7246] etc., these will be covered by future Internet Drafts.

2.2. Optional capabilities

This model is designed to represent the capabilities of MVPN devices with various specifications, including some with basic subsets of the MVPN protocols. The main design goals of this draft are that any major now-existing implementation may be said to support the basic model, and that the configuration of all implementations meeting the specification is easy to express through some combination of the features in the basic model and simple vendor augmentations.

On the other hand, operational state parameters are not so widely designated as features, as there are many cases where the defaulting of an operational state parameter would not cause any harm to the system, and it is much more likely that an implementation without native support for a piece of operational state would be able to

derive a suitable value for a state variable that is not natively supported.

For the same reason, wide constant ranges (for example, timer maximum and minimum) will be used in the model. It is expected that vendors will augment the model with any specific restrictions that might be required. Vendors may also extend the features list with proprietary extensions.

2.3. Position of address family in hierarchy

The current draft contains MVPN ipv4 and ipv6 as separate schema branches in the structure. The reason for this is to inherit l3vpn yang model structure and make it easier for implementations which may optionally choose to support specific address families. And the names of objects may be different between the ipv4 and ipv6 address families.

3. Module Structure

3.1. MVPN Configuration

The MVPN modules define the network-instance-wide configuration options in a two-level hierarchy as listed below:

Instance level: MVPN configuration attributes for the entire routing instance, including route-target, I-PMSI tunnel and S-PMSI number, common timer etc.

PMSI tunnel level: MVPN configuration attributes applicable to the I-PMSI and per S-PMSI tunnel configuration attributes, including tunnel mode, tunnel specific parameters and threshold etc.

Where fields are not genuinely essential to protocol operation, they are marked as optional. Some fields will be essential but have a default specified, so that they need not be configured explicitly. The module structure also applies, where applicable, to the operational state as well.

Our current direction is to agree to a network-instance-centric (VRF) model as opposed to protocol-centric mainly because it inherits l3vpn and it can as a part of l3vpn yang model. It fits well into the routing-instance model, and it is easier to map from the VRF-centric to the protocol-centric than the other way around due to forward references.

The MVPN model will augment "/ni:network-instances/ni:network-instance/l3vpn:".

```
   augment /ni:network-instances/ni:network-instance:
      +--rw l3vpn
         +--rw ipv4
         |  +--rw mvpn
         |  |  +--rw signaling-mode?        enumeration
         |  |  +--rw auto-discovery-mode?   enumeration
         |  |  +--rw config-type?           enumeration
         |  |  +--rw is-sender-site?        boolean
         |  |  +--rw rpt-spt-mode?          boolean
         |  |  +--rw mvpn-route-targets
         |  |  |  +--rw mvpn-route-target* [rt-type rt-value]
         |  |  |     +--rw rt-type     enumeration
         |  |  |     +--rw rt-value    string
         |  |  +--rw mvpn-ipmsi-tunnel
         |  |  |  +--rw tunnel-mode?                enumeration
         |  |  |  +--rw (ipmsi-type)?
         |  |  |     +--:(p2mp-te)
         |  |  |     |  +--rw te-p2mp-template?           string
         |  |  |     +--:(p2mp-mldp)
         |  |  |     +--:(pim-ssm)
         |  |  |     |  +--rw ssm-default-group-addr?     inet:ip-address
         |  |  |     +--:(pim-sm)
         |  |  |     |  +--rw sm-default-group-addr?      inet:ip-address
         |  |  |     +--:(bidir-pim)
         |  |  |     |  +--rw bidir-default-group-addr?   inet:ip-address
         |  |  |     +--:(pim-dm)
         |  |  |     |  +--rw dm-default-group-addr?      inet:ip-address
         |  |  |     +--:(ingress-replication)
         |  |  |     +--:(mp2mp-mldp)
         |  |  +--rw mvpn-spmsi-tunnels
         |  |     +--rw switch-delay-time?   uint8
         |  |     +--rw hold-down-time?      uint16
         |  |     +--rw tunnel-limit?        uint16
         |  |     +--rw mvpn-spmsi-tunnel* [tunnel-mode]
         |  |        +--rw tunnel-mode                    enumeration
         |  |        +--rw (spmsi-type)?
         |  |        |  +--:(p2mp-te)
         |  |        |  |  +--rw te-p2mp-template?              string
         |  |        |  +--:(p2mp-mldp)
         |  |        |  +--:(pim-ssm)
         |  |        |  |  +--rw ssm-group-pool-addr?           inet:ip-address
         |  |        |  |  +--rw ssm-group-pool-masklength?     uint8
         |  |        |  +--:(pim-sm)
         |  |        |  |  +--rw sm-group-pool-addr?            inet:ip-address
         |  |        |  |  +--rw sm-group-pool-masklength?      uint8
         |  |        |  +--:(bidir-pim)
         |  |        |  |  +--rw bidir-group-pool-addr?         inet:ip-address
         |  |        |  |  +--rw bidir-group-pool-masklength?   uint8
         |  |        |  +--:(pim-dm)
```

```
         |  |        |  |  +--rw dm-group-pool-addr?            inet:ip-address
         |  |        |  |  +--rw dm-group-pool-masklength?      uint8
         |  |        |  +--:(ingress-replication)
         |  |        |  +--:(mp2mp-mldp)
         |  |        +--rw switch-threshold?              uint32
         |  |        +--rw (address-mask-or-acl)?
         |  |           +--:(address-mask)
         |  |           |  +--rw ipv4-group-addr?               inet:ipv4-addres
s
         |  |           |  +--rw ipv4-group-masklength?         uint8
         |  |           |  +--rw ipv4-source-addr?              inet:ipv4-addres
s
         |  |           |  +--rw ipv4-source-masklength?        uint8
         |  |           +--:(acl)
         |  |              +--rw group-acl-ipv4?                string
         |  +--ro mvpn-state
         |
         +--rw ipv6
            +--rw mvpn
               |  +--rw signaling-mode?        enumeration
               |  +--rw auto-discovery-mode?   enumeration
               |  +--rw config-type?           enumeration
               |  +--rw is-sender-site?        boolean
               |  +--rw rpt-spt-mode?          boolean
               |  +--rw mvpn-route-targets
               |  |  +--rw mvpn-route-target* [rt-type rt-value]
               |  |     +--rw rt-type     enumeration
               |  |     +--rw rt-value    string
               |  +--rw mvpn-ipmsi-tunnel
               |  |  +--rw tunnel-mode?               enumeration
               |  |  +--rw (ipmsi-type)?
               |  |     +--:(p2mp-te)
               |  |     |  +--rw te-p2mp-template?           string
               |  |     +--:(p2mp-mldp)
               |  |     +--:(pim-ssm)
               |  |     |  +--rw ssm-default-group-addr?     inet:ip-address
               |  |     +--:(pim-sm)
               |  |     |  +--rw sm-default-group-addr?      inet:ip-address
               |  |     +--:(bidir-pim)
               |  |     |  +--rw bidir-default-group-addr?   inet:ip-address
               |  |     +--:(pim-dm)
               |  |     |  +--rw dm-default-group-addr?      inet:ip-address
               |  |     +--:(ingress-replication)
               |  |     +--:(mp2mp-mldp)
               |  +--rw mvpn-spmsi-tunnels
               |     +--rw switch-delay-time?   uint8
               |     +--rw hold-down-time?      uint16
               |     +--rw tunnel-limit?        uint16
               |     +--rw mvpn-spmsi-tunnel* [tunnel-mode]
               |        +--rw tunnel-mode                   enumeration
               |        +--rw (spmsi-type)?
```

```
            |        |  +--:(p2mp-te)
            |        |  |  +--rw te-p2mp-template?              string
            |        |  +--:(p2mp-mldp)
            |        |  +--:(pim-ssm)
            |        |  |  +--rw ssm-group-pool-addr?           inet:ip-address
            |        |  |  +--rw ssm-group-pool-masklength?     uint8
            |        |  +--:(pim-sm)
            |        |  |  +--rw sm-group-pool-addr?            inet:ip-address
            |        |  |  +--rw sm-group-pool-masklength?      uint8
            |        |  +--:(bidir-pim)
            |        |  |  +--rw bidir-group-pool-addr?         inet:ip-address
            |        |  |  +--rw bidir-group-pool-masklength?   uint8
            |        |  +--:(pim-dm)
            |        |  |  +--rw dm-group-pool-addr?            inet:ip-address
            |        |  |  +--rw dm-group-pool-masklength?      uint8
            |        |  +--:(ingress-replication)
            |        |  +--:(mp2mp-mldp)
            |        +--rw switch-threshold?              uint32
            |        +--rw (address-mask-or-acl)?
            |           +--:(address-mask)
            |           |  +--rw ipv6-group-addr?               inet:ipv6-addres
s
            |           |  +--rw ipv6-groupmasklength?          uint8
            |           |  +--rw ipv6-source-addr?              inet:ipv6-addres
s
            |           |  +--rw ipv6-source-masklength?        uint8
            |           +--:(acl)
            |              +--rw group-acl-ipv6?                string
            +--ro mvpn-state
```

3.2. MVPN Operational State

The MVPN module contains operational state information also in a two-level hierarchy as mentioned earlier.

Instance level: MVPN operational state attributes for the entire routing instance, including route-target, I-PMSI tunnel and S-PMSI number, common timer etc.

PMSI tunnel level: MVPN PMSI tunnel operational state attributes applicable to the I-PMSI and per S-PMSI tunnel operational state attributes, including tunnel mode, tunnel role, tunnel specific parameters and referenced private source and group address etc.

```
augment /ni:network-instances/ni:network-instance:
   +--rw l3vpn
      +--rw ipv4
      |  +--rw mvpn
      |  |
      |  +--ro mvpn-state
```

```
|     +--ro signaling-mode?                 enumeration
|     +--ro auto-discovery-mode?            enumeration
|     +--ro config-type?                    enumeration
|     +--ro is-sender-site?                 boolean
|     +--ro rpt-spt-mode?                   boolean
|     +--ro mvpn-route-targets
|     |  +--ro mvpn-route-target* [rt-type rt-value]
|     |     +--ro rt-type     enumeration
|     |     +--ro rt-value    string
|     +--ro mvpn-ipmsi-tunnel
|     |  +--ro tunnel-mode?                enumeration
|     |  +--ro (ipmsi-type)?
|     |     +--:(p2mp-te)
|     |     |  +--ro te-p2mp-template?           string
|     |     +--:(p2mp-mldp)
|     |     +--:(pim-ssm)
|     |     |  +--ro ssm-default-group-addr?     inet:ip-address
|     |     +--:(pim-sm)
|     |     |  +--ro sm-default-group-addr?      inet:ip-address
|     |     +--:(bidir-pim)
|     |     |  +--ro bidir-default-group-addr?   inet:ip-address
|     |     +--:(pim-dm)
|     |     |  +--ro dm-default-group-addr?      inet:ip-address
|     |     +--:(ingress-replication)
|     |     +--:(mp2mp-mldp)
|     +--ro mvpn-spmsi-tunnels
|     |  +--ro switch-delay-time?   uint8
|     |  +--ro hold-down-time?      uint16
|     |  +--ro tunnel-limit?        uint16
|     |  +--ro mvpn-spmsi-tunnel* [tunnel-mode]
|     |     +--ro tunnel-mode                   enumeration
|     |     +--ro (spmsi-type)?
|     |     |  +--:(p2mp-te)
|     |     |  |  +--ro te-p2mp-template?              string
|     |     |  +--:(p2mp-mldp)
|     |     |  +--:(pim-ssm)
|     |     |  |  +--ro ssm-group-pool-addr?           inet:ip-address
|     |     |  |  +--ro ssm-group-pool-masklength?     uint8
|     |     |  +--:(pim-sm)
|     |     |  |  +--ro sm-group-pool-addr?            inet:ip-address
|     |     |  |  +--ro sm-group-pool-masklength?      uint8
|     |     |  +--:(bidir-pim)
|     |     |  |  +--ro bidir-group-pool-addr?         inet:ip-address
|     |     |  |  +--ro bidir-group-pool-masklength?   uint8
|     |     |  +--:(pim-dm)
|     |     |  |  +--ro dm-group-pool-addr?            inet:ip-address
|     |     |  |  +--ro dm-group-pool-masklength?      uint8
|     |     |  +--:(ingress-replication)
|     |     |  +--:(mp2mp-mldp)
```

```
          |     |     +--ro switch-threshold?             uint32
          |     |     +--ro (address-mask-or-acl)?
          |     |        +--:(address-mask)
          |     |        |  +--ro ipv4-group-addr?              inet:ipv4-addres
s
          |     |        |  +--ro ipv4-group-masklength?        uint8
          |     |        |  +--ro ipv4-source-addr?             inet:ipv4-addres
s
          |     |        |  +--ro ipv4-source-masklength?       uint8
          |     |        +--:(acl)
          |     |           +--ro group-acl-ipv4?              string
          |     +--ro mvpn-ipmsi-tunnel-info
          |     |  +--ro tunnel-mode?             enumeration
          |     |  +--ro (pmsi-type)?
          |     |  |  +--:(p2mp-te)
          |     |  |  |  +--ro te-p2mp-id?              uint16
          |     |  |  |  +--ro te-tunnel-id?            uint16
          |     |  |  |  +--ro te-extend-tunnel-id?     uint16
          |     |  |  +--:(p2mp-mldp)
          |     |  |  |  +--ro mldp-root-addr?          inet:ip-address
          |     |  |  |  +--ro mldp-lsp-id?             string
          |     |  |  +--:(pim-ssm)
          |     |  |  |  +--ro ssm-group-addr?          inet:ip-address
          |     |  |  +--:(pim-sm)
          |     |  |  |  +--ro sm-group-addr?           inet:ip-address
          |     |  |  +--:(bidir-pim)
          |     |  |  |  +--ro bidir-group-addr?        inet:ip-address
          |     |  |  +--:(pim-dm)
          |     |  |  |  +--ro dm-group-addr?           inet:ip-address
          |     |  |  +--:(ingress-replication)
          |     |  |  +--:(mp2mp-mldp)
          |     |  +--ro tunnel-role?             enumeration
          |     |  +--ro mvpn-pmsi-sg-ref-ipv4s
          |     |     +--ro mvpn-pmsi-sg-ref-ipv4* [ipv4-source-address ipv4-grou
p-address]
          |     |        +--ro ipv4-source-address    inet:ipv4-address
          |     |        +--ro ipv4-group-address     inet:ipv4-address
          |     +--ro mvpn-spmsi-tunnel-ipv4-infos
          |        +--ro mvpn-spmsi-tunnel-ipv4-info* [tunnel-mode]
          |           +--ro tunnel-mode               enumeration
          |           +--ro (pmsi-type)?
          |           |  +--:(p2mp-te)
          |           |  |  +--ro te-p2mp-id?              uint16
          |           |  |  +--ro te-tunnel-id?            uint16
          |           |  |  +--ro te-extend-tunnel-id?     uint16
          |           |  +--:(p2mp-mldp)
          |           |  |  +--ro mldp-root-addr?          inet:ip-address
          |           |  |  +--ro mldp-lsp-id?             string
          |           |  +--:(pim-ssm)
          |           |  |  +--ro ssm-group-addr?          inet:ip-address
          |           |  +--:(pim-sm)
          |           |  |  +--ro sm-group-addr?           inet:ip-address
```

```
         |           |  +--:(bidir-pim)
         |           |  |  +--ro bidir-group-addr?         inet:ip-address
         |           |  +--:(pim-dm)
         |           |  |  +--ro dm-group-addr?            inet:ip-address
         |           |  +--:(ingress-replication)
         |           |  +--:(mp2mp-mldp)
         |           +--ro tunnel-role?              enumeration
         |           +--ro mvpn-pmsi-sg-ref-ipv4s
         |              +--ro mvpn-pmsi-sg-ref-ipv4* [ipv4-source-address ipv4-g
roup-address]
         |                 +--ro ipv4-source-address    inet:ipv4-address
         |                 +--ro ipv4-group-address     inet:ipv4-address
         +--rw ipv6
            +--rw mvpn
            |
            +--ro mvpn-state
               +--ro signaling-mode?                 enumeration
               +--ro auto-discovery-mode?            enumeration
               +--ro config-type?                    enumeration
               +--ro is-sender-site?                 boolean
               +--ro rpt-spt-mode?                   boolean
               +--ro mvpn-route-targets
               |  +--ro mvpn-route-target* [rt-type rt-value]
               |     +--ro rt-type     enumeration
               |     +--ro rt-value    string
               +--ro mvpn-ipmsi-tunnel
               |  +--ro tunnel-mode?                enumeration
               |  +--ro (ipmsi-type)?
               |     +--:(p2mp-te)
               |     |  +--ro te-p2mp-template?           string
               |     +--:(p2mp-mldp)
               |     +--:(pim-ssm)
               |     |  +--ro ssm-default-group-addr?     inet:ip-address
               |     +--:(pim-sm)
               |     |  +--ro sm-default-group-addr?      inet:ip-address
               |     +--:(bidir-pim)
               |     |  +--ro bidir-default-group-addr?   inet:ip-address
               |     +--:(pim-dm)
               |     |  +--ro dm-default-group-addr?      inet:ip-address
               |     +--:(ingress-replication)
               |     +--:(mp2mp-mldp)
               +--ro mvpn-spmsi-tunnels
               |  +--ro switch-delay-time?   uint8
               |  +--ro hold-down-time?      uint16
               |  +--ro tunnel-limit?        uint16
               |  +--ro mvpn-spmsi-tunnel* [tunnel-mode]
               |     +--ro tunnel-mode                    enumeration
               |     +--ro (spmsi-type)?
               |     |  +--:(p2mp-te)
               |     |  |  +--ro te-p2mp-template?             string
```

```
                |     |  +--:(p2mp-mldp)
                |     |  +--:(pim-ssm)
                |     |  |  +--ro ssm-group-pool-addr?           inet:ip-address
                |     |  |  +--ro ssm-group-pool-masklength?     uint8
                |     |  +--:(pim-sm)
                |     |  |  +--ro sm-group-pool-addr?            inet:ip-address
                |     |  |  +--ro sm-group-pool-masklength?      uint8
                |     |  +--:(bidir-pim)
                |     |  |  +--ro bidir-group-pool-addr?         inet:ip-address
                |     |  |  +--ro bidir-group-pool-masklength?   uint8
                |     |  +--:(pim-dm)
                |     |  |  +--ro dm-group-pool-addr?            inet:ip-address
                |     |  |  +--ro dm-group-pool-masklength?      uint8
                |     |  +--:(ingress-replication)
                |     |  +--:(mp2mp-mldp)
                |     +--ro switch-threshold?              uint32
                |     +--ro (address-mask-or-acl)?
                |        +--:(address-mask)
                |        |  +--ro ipv6-group-addr?               inet:ipv6-addres
s
                |        |  +--ro ipv6-groupmasklength?          uint8
                |        |  +--ro ipv6-source-addr?              inet:ipv6-addres
s
                |        |  +--ro ipv6-source-masklength?        uint8
                |        +--:(acl)
                |           +--ro group-acl-ipv6?                string
                +--ro mvpn-ipmsi-tunnel-info
                |  +--ro tunnel-mode?              enumeration
                |  +--ro (pmsi-type)?
                |  |  +--:(p2mp-te)
                |  |  |  +--ro te-p2mp-id?               uint16
                |  |  |  +--ro te-tunnel-id?             uint16
                |  |  |  +--ro te-extend-tunnel-id?      uint16
                |  |  +--:(p2mp-mldp)
                |  |  |  +--ro mldp-root-addr?           inet:ip-address
                |  |  |  +--ro mldp-lsp-id?              string
                |  |  +--:(pim-ssm)
                |  |  |  +--ro ssm-group-addr?           inet:ip-address
                |  |  +--:(pim-sm)
                |  |  |  +--ro sm-group-addr?            inet:ip-address
                |  |  +--:(bidir-pim)
                |  |  |  +--ro bidir-group-addr?         inet:ip-address
                |  |  +--:(pim-dm)
                |  |  |  +--ro dm-group-addr?            inet:ip-address
                |  |  +--:(ingress-replication)
                |  |  +--:(mp2mp-mldp)
                |  +--ro tunnel-role?              enumeration
                |  +--ro mvpn-pmsi-sg-ref-ipv6s
                |     +--ro mvpn-pmsi-sg-ref-ipv6* [ipv6-source-address ipv6-grou
p-address]
                |        +--ro ipv6-source-address    inet:ipv6-address
                |        +--ro ipv6-group-address     inet:ipv6-address
```

```
               +--ro mvpn-spmsi-tunnel-ipv6-infos
                  +--ro mvpn-spmsi-tunnel-ipv6-info* [tunnel-mode]
                     +--ro tunnel-mode               enumeration
                     +--ro (pmsi-type)?
                     |  +--:(p2mp-te)
                     |  |  +--ro te-p2mp-id?              uint16
                     |  |  +--ro te-tunnel-id?            uint16
                     |  |  +--ro te-extend-tunnel-id?     uint16
                     |  +--:(p2mp-mldp)
                     |  |  +--ro mldp-root-addr?          inet:ip-address
                     |  |  +--ro mldp-lsp-id?             string
                     |  +--:(pim-ssm)
                     |  |  +--ro ssm-group-addr?          inet:ip-address
                     |  +--:(pim-sm)
                     |  |  +--ro sm-group-addr?           inet:ip-address
                     |  +--:(bidir-pim)
                     |  |  +--ro bidir-group-addr?        inet:ip-address
                     |  +--:(pim-dm)
                     |  |  +--ro dm-group-addr?           inet:ip-address
                     |  +--:(ingress-replication)
                     |  +--:(mp2mp-mldp)
                     +--ro tunnel-role?              enumeration
                     +--ro mvpn-pmsi-sg-ref-ipv6s
                        +--ro mvpn-pmsi-sg-ref-ipv6* [ipv6-source-address ipv6-g
roup-address]
                           +--ro ipv6-source-address    inet:ipv6-address
                           +--ro ipv6-group-address     inet:ipv6-address
```

4. MVPN YANG Modules

```
   <CODE BEGINS> file "ietf-mvpn@2017-03-07.yang"
   module ietf-mvpn {
      namespace "urn:ietf:params:xml:ns:yang:ietf-mvpn";
      prefix mvpn;

      import ietf-network-instance {
       prefix ni;
      }

      import ietf-inet-types {
        prefix inet;
      }


      organization
        "IETF BESS(BGP Enabled Services) Working Group";
      contact
        "liuyisong@huawei.com
         guofeng@huawei.com
```

```
         Xufeng_Liu@jabil.com
         rkebler@juniper.net
         masivaku@cisco.com";
      description
        "This YANG module defines the generic configuration
         data for mvpn, which is common across all of the vendor
         implementations of the protocol. It is intended that the module
         will be extended by vendors to define vendor-specific
         mvpn configuration parameters.";

      revision 2017-03-07 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: A YANG Data Model for MVPN";
      }

      grouping mvpn-instance-config {
        description "Mvpn basic configuration per instance.";

        leaf signaling-mode {
          type enumeration {
            enum invalid {
              value "0";
              description "invalid";
            }
            enum bgp {
              value "1";
              description "bgp";
            }
            enum pim {
              value "2";
              description "pim";
            }
          }
          default "invalid";
          description "Signaling mode.";
        }
        leaf auto-discovery-mode {
          type enumeration {
            enum none {
              value "0";
              description "none";
            }
            enum ad {
              value "1";
              description "auto-discovery";
            }
          }
```

```
          default "none";
          description "Auto discovery mode.";
        }
        leaf config-type {
          type enumeration {
            enum md {
              value "0";
              description "md";
            }
            enum ng {
              value "1";
              description "ng";
            }
          }
          default "md";
          description "Mvpn type, which can be md(rosen) mvpn or ng mvpn.";
        }
        leaf is-sender-site {
          type boolean;
          default "false";
          description "Configure the current PE as a sender PE.";
        }
        leaf rpt-spt-mode {
          type boolean;
          default "false";
          description "Rpt and spt mode in multicast private net.";
        }

      }

      grouping mvpn-vpn-targets {
        description "May be different from l3vpn unicast route-targets";
        container mvpn-route-targets{
          description "Multicast vpn route-targets";
          list mvpn-route-target {
            key "rt-type rt-value" ;
            description
              "List of multicast route-targets" ;
            leaf rt-type {
               type enumeration {
                 enum export-extcommunity {
                   value "0";
                   description "export-extcommunity";
                 }
                 enum import-extcommunity {
                   value "1";
                   description "import-extcommunity";
                 }
               }
```

```
              mandatory "true";
              description
                "rt types are as follows:
                 export-extcommunity: specifies the value of
                 the extended community attribute of the
                 route from an outbound interface to the
                 destination vpn.
                 import-extcommunity: receives routes that
                 carry the specified extended community
                 attribute";
            }
            leaf rt-value {
              type string {
                length "3..21";
              }
              description
                "the available mvpn target formats are as
                 follows:
                 - 16-bit as number:32-bit user-defined
                 number, for example, 1:3. an as number
                 ranges from 0 to 65535, and a user-defined
                 number ranges from 0 to 4294967295. The as
                 number and user-defined number cannot be
                 both 0s. That is, a vpn target cannot be 0:0.
                 - 32-bit ip address:16-bit user-defined
                 number, for example, 192.168.122.15:1.
                 The ip address ranges from 0.0.0.0 to
                 255.255.255.255, and the user-defined
                 number ranges from 0 to 65535.";
            }
          }
        }
      }


    grouping mvpn-ipmsi-tunnel-config {
      description "Default mdt for rosen mvpn and I-PMSI for ng mvpn";

      container mvpn-ipmsi-tunnel {
        description "I-PMSI tunnel configuraton";
        leaf tunnel-mode {
          type enumeration {
            enum invalid {
              value "0";
              description "invalid";
            }
            enum p2mp-te {
              value "1";
```

```
            description "p2mp-te";
          }
          enum p2mp-mldp {
            value "2";
            description "p2mp-mldp";
          }
          enum pim-ssm {
            value "3";
            description "pim-ssm";
          }
          enum pim-sm {
            value "4";
            description "pim-sm";
          }
          enum bidir-pim {
            value "5";
            description "bidir-pim";
          }
          enum ingress-replication {
            value "6";
            description "ingress-replication";
          }
          enum mp2mp-mldp {
            value "7";
            description "mp2mp-mldp";
          }
          enum pim-dm {
            value "8";
            description "pim-dm";
          }
        }
        description "I-PMSI tunnel mode.";
      }
      choice ipmsi-type {
        description "I-PMSI tunnel parameter configuration";
        case p2mp-te {
          description "P2mp TE tunnel";
          leaf te-p2mp-template {
            type string {
              length "1..31";
            }
            description "P2mp te tunnel template";
          }
        }
        case p2mp-mldp {
          description "Mldp tunnel";
        }
        case pim-ssm {
          description "Pim ssm tunnel";
```

```
            leaf ssm-default-group-addr {
              type inet:ip-address;
              description "Default mdt or I-PMSI group address.";
            }
          }
          case pim-sm {
            description "Pim sm tunnel";
            leaf sm-default-group-addr {
              type inet:ip-address;
              description "Default mdt or I-PMSI group address.";
            }
          }
          case bidir-pim {
            description "Bidir pim tunnel";
            leaf bidir-default-group-addr {
              type inet:ip-address;
              description "Default mdt or I-PMSI group address.";
            }
          }
          case pim-dm {
            description "Pim dm tunnel";
            leaf dm-default-group-addr {
              type inet:ip-address;
              description "Default mdt or I-PMSI group address.";
            }
          }
          case ingress-replication {
            description "Ingress replication p2p tunnel";
          }
          case mp2mp-mldp {
            description "Mp2mp mldp tunnel";
          }
        }
      }
    }

    grouping mvpn-spmsi-tunnel-basic-config {
      description "S-PMSI tunnel basic configuration";
      leaf tunnel-mode {
        type enumeration {
          enum invalid {
            value "0";
            description "invalid";
          }
          enum p2mp-te {
            value "1";
            description "p2mp-te";
          }
          enum p2mp-mldp {
```

```
              value "2";
              description "p2mp-mldp";
            }
            enum pim-ssm {
              value "3";
              description "pim-ssm";
            }
            enum pim-sm {
              value "4";
              description "pim-sm";
            }
            enum bidir-pim {
              value "5";
              description "bidir-pim";
            }
            enum ingress-replication {
              value "6";
              description "ingress-replication";
            }
            enum mp2mp-mldp {
              value "7";
              description "mp2mp-mldp";
            }
            enum pim-dm {
              value "8";
              description "pim-dm";
            }
          }
          description "S-PMSI tunnel mode.";
        }
        choice spmsi-type {
          description "S-PMSI tunnel parameter configuration";
          case p2mp-te {
            description "P2mp te tunnel";
            leaf te-p2mp-template {
              type string {
                length "1..31";
              }
              description "P2mp te tunnel template";
            }
          }
          case p2mp-mldp {
            description "Mldp tunnel";
          }
          case pim-ssm {
            description "Pim ssm tunnel";
            leaf ssm-group-pool-addr {
              type inet:ip-address;
              description "Group pool address for data mdt or pim s-pmsi.";
```

```
            }
            leaf ssm-group-pool-masklength {
              type uint8 {
                range "8..128";
              }
              description "Group pool mask for data mdt or pim s-pmsi";
            }
          }
          case pim-sm {
            description "Pim sm tunnel";
            leaf sm-group-pool-addr {
              type inet:ip-address;
              description "Group pool address for data mdt or pim s-pmsi.";
            }
            leaf sm-group-pool-masklength {
              type uint8 {
                range "8..128";
              }
              description "Group pool mask for data mdt or pim s-pmsi";
            }
          }
          case bidir-pim {
            description "Bidir pim tunnel";
            leaf bidir-group-pool-addr {
              type inet:ip-address;
              description "Group pool address for data mdt or pim s-pmsi.";
            }
            leaf bidir-group-pool-masklength {
              type uint8 {
                range "8..128";
              }
              description "Group pool mask for data mdt or pim s-pmsi";
            }
          }
          case pim-dm {
            description "Pim dm tunnel";
            leaf dm-group-pool-addr {
              type inet:ip-address;
              description "Group pool address for data mdt or pim s-pmsi.";
            }
            leaf dm-group-pool-masklength {
              type uint8 {
                range "8..128";
              }
              description "Group pool mask for data mdt or pim s-pmsi";
            }
          }
          case ingress-replication {
            description "Ingress replication p2p tunnel";
```

```
          }
          case mp2mp-mldp {
            description "Mp2mp mldp tunnel";
          }
        }
        leaf switch-threshold {
          type uint32 {
            range "0..4194304";
          }
          default "0";
          description
            "Multicast packet rate threshold for
             triggering the switching from the
             I-PMSI to the S-PMSI. The value is
             an integer ranging from 0 to 4194304, in
             kbit/s. The default value is 0.";
        }
      }

      grouping mvpn-spmsi-tunnel-config-ipv4 {
        description "Data mdt for rosen mvpn and S-PMSI for ng mvpn";

        container mvpn-spmsi-tunnels {
          description "S-PMSI tunnel configuration";
          leaf switch-delay-time {
            type uint8 {
              range "3..60";
            }
            default "5";
            description
            "Delay for switching from the I-PMSI to
             the S-PMSI. The value is an integer
             ranging from 3 to 60, in seconds. ";
          }
          leaf hold-down-time {
            type uint16 {
              range "0..512";
            }
            default "60";
            description
              "Delay for switching back from the S-PMSI
               to the I-PMSI. The value is an integer
               ranging from 0 to 512, in seconds. ";
          }
          leaf tunnel-limit {
            type uint16 {
              range "1..1024";
            }
            description
```

```
            "Maximum number of s-pmsi tunnels allowed.";
          }

          list mvpn-spmsi-tunnel {
            key "tunnel-mode";
            description "S-PMSI tunnel parameter configuration";

            uses mvpn-spmsi-tunnel-basic-config;

            choice address-mask-or-acl {
              description "Type of define private net multicast address range";
              case address-mask {
                description "Use the type of address and mask";
                leaf ipv4-group-addr {
                  type inet:ipv4-address;
                  description
                    "Start and end ipv4 addresses of the group
                     address in private net. ";
                }
                leaf ipv4-group-masklength {
                  type uint8 {
                    range "4..32";
                  }
                  description
                    "Group mask length for ipv4 addresses in
                     the group address pool in private net.";
                }
                leaf ipv4-source-addr {
                  type inet:ipv4-address;
                  description
                    "Start and end ipv4 addresses of the source
                     address in private net.";
                }
                leaf ipv4-source-masklength {
                  type uint8 {
                    range "0..32";
                  }
                  description
                    "Source mask length for ipv4 addresses in
                     the group address pool in private net.";
                }
              }
              case acl {
                description "Use the type of acl";
                leaf group-acl-ipv4 {
                  type string {
                    length "1..32";
                  }
                  description
```

```
                    "Specify the (s, g) entry on which the
                     S-PMSI tunnel takes effect.
                     The value is an integer ranging from 3000
                     to 3999 or a string of 32 case-sensitive
                     characters. If no value is specified, the
                     switch-group address pool takes effect on
                     all (s, g).";
                }
              }
            }
          }
        }
      }

      grouping mvpn-spmsi-tunnel-config-ipv6 {
        description "Data mdt for rosen mvpn and S-PMSI for ng mvpn";

        container mvpn-spmsi-tunnels {
          description "S-PMSI tunnel configuration";
          leaf switch-delay-time {
            type uint8 {
              range "3..60";
            }
            default "5";
            description
            "Delay for switching from the I-PMSI to
             the S-PMSI. The value is an integer
             ranging from 3 to 60, in seconds. ";
          }
          leaf hold-down-time {
            type uint16 {
              range "0..512";
            }
            default "60";
            description
              "Delay for switching back from the S-PMSI
               to the I-PMSI. The value is an integer
               ranging from 0 to 512, in seconds. ";
          }
          leaf tunnel-limit {
            type uint16 {
              range "1..1024";
            }
            description
              "Maximum number of s-pmsi tunnels allowed.";
          }

          list mvpn-spmsi-tunnel {
            key "tunnel-mode";
```

```
            description "S-PMSI tunnel parameter configuration";

            uses mvpn-spmsi-tunnel-basic-config;

            choice address-mask-or-acl {
              description "Type of define private net multicast address range";
              case address-mask {
                description "Use the type of address and mask";
                leaf ipv6-group-addr {
                  type inet:ipv6-address;
                  description
                    "Start and end ipv6 addresses of the group
                     address in private net.";
                }
                leaf ipv6-groupmasklength {
                  type uint8 {
                    range "8..128";
                  }
                  description
                    "Group mask length for ipv6 addresses in
                     the group address pool in private net.";
                }
                leaf ipv6-source-addr {
                  type inet:ipv6-address;
                  description
                    "Start and end ipv6 addresses of the source
                     address in private net.";
                }
                leaf ipv6-source-masklength {
                  type uint8 {
                    range "0..128";
                  }
                  description
                    "Source mask length for ipv6 addresses in
                     the group address pool in private net.";
                }
              }
              case acl {
                description "Use the type of acl";
                leaf group-acl-ipv6 {
                  type string {
                    length "1..32";
                  }
                  description
                    "Specify the (s, g) entry on which the
                     S-PMSI tunnel takes effect.
                     The value is an integer ranging from 3000
                     to 3999 or a string of 32 case-sensitive
                     characters. If no value is specified, the
```

```
                    switch-group address pool takes effect on
                    all (s, g).";
                }
              }
            }
          }
        }
      }

      grouping mvpn-pmsi-state {
        description "PMSI tunnel operational state information";
        leaf tunnel-mode {
          type enumeration {
            enum invalid {
              value "0";
              description "invalid";
            }
            enum p2mp-te {
              value "1";
              description "p2mp-te";
            }
            enum p2mp-mldp {
              value "2";
              description "p2mp-mldp";
            }
            enum pim-ssm {
              value "3";
              description "pim-ssm";
            }
            enum pim-sm {
              value "4";
              description "pim-sm";
            }
            enum bidir-pim {
              value "5";
              description "bidir-pim";
            }
            enum ingress-replication {
              value "6";
              description "ingress-replication";
            }
            enum mp2mp-mldp {
              value "7";
              description "mp2mp-mldp";
            }
            enum pim-dm {
              value "8";
              description "pim-dm";
            }
```

```
          }
          description "PMSI tunnel mode.";
        }
        choice pmsi-type {
          description "PMSI tunnel operational state information for each type";
          case p2mp-te {
            description "P2mp te tunnel";
            leaf te-p2mp-id {
              type uint16 {
                range "0..65535";
              }
              default "0";
              description "P2mp id of the p2mp tunnel.";
            }
            leaf te-tunnel-id {
              type uint16 {
                range "1..65535";
              }
              description "Id of the p2mp tunnel.";
            }
            leaf te-extend-tunnel-id {
              type uint16 {
                range "1..65535";
              }
              description "P2mp extended tunnel interface id.";
            }
          }
          case p2mp-mldp {
            description "P2mp mldp tunnel";
            leaf mldp-root-addr {
              type inet:ip-address;
              description "Ip address of the root of a p2mp ldp lsp.";
            }
            leaf mldp-lsp-id {
              type string {
                length "1..256";
              }
              description "P2mp ldp lsp id.";
            }
          }
          case pim-ssm {
            description "Pim ssm tunnel";
            leaf ssm-group-addr {
              type inet:ip-address;
              description "Group address for pim ssm";
            }
          }
          case pim-sm {
            description "Pim sm tunnel";
```

```
            leaf sm-group-addr {
              type inet:ip-address;
              description "Group address for pim sm";
            }
          }
          case bidir-pim {
            description "Bidir pim tunnel";
            leaf bidir-group-addr {
              type inet:ip-address;
              description "Group address for bidir-pim";
            }
          }
          case pim-dm {
            description "Pim dm tunnel";
            leaf dm-group-addr {
              type inet:ip-address;
              description "Group address for pim-dm";
            }
          }
          case ingress-replication {
            description "Ingress replication p2p tunnel";
          }
          case mp2mp-mldp {
            description "mp2mp mldp tunnel";
          }
        }
        leaf tunnel-role {
          type enumeration {
            enum none {
              value "0";
              description "none";
            }
            enum root {
              value "1";
              description "root";
            }
            enum leaf {
              value "2";
              description "leaf";
            }
            enum root-and-leaf {
              value "3";
              description "root-and-leaf";
            }
          }
          description "Role of a tunnel node.";
        }
      }
```

```
      grouping mvpn-pmsi-entry-ipv4 {
        description
          "Multicast entries in ipv4 mvpn referenced the pmsi tunnel";
        container mvpn-pmsi-sg-ref-ipv4s {
          description
            "Multicast entries in ipv4 mvpn referenced the pmsi tunnel";
          list mvpn-pmsi-sg-ref-ipv4 {
            key "ipv4-source-address ipv4-group-address";
            description "Ipv4 source and group address";
            leaf ipv4-source-address {
              type inet:ipv4-address;
              description "Source address in I-PMSI or S-PMSI for ipv4.";
            }
            leaf ipv4-group-address {
              type inet:ipv4-address;
              description "Group address in I-PMSI or S-PMSI for ipv4.";
            }
          }
        }
      }


      grouping mvpn-pmsi-entry-ipv6 {
        description
          "Multicast entries in ipv6 mvpn referenced the pmsi tunnel";
        container mvpn-pmsi-sg-ref-ipv6s {
          description
            "Multicast entries in ipv6 mvpn referenced the pmsi tunnel";
          list mvpn-pmsi-sg-ref-ipv6 {
            key "ipv6-source-address ipv6-group-address";
            description "Ipv6 source and group address";
            leaf ipv6-source-address {
              type inet:ipv6-address;
              description "Source address in I-PMSI or S-PMSI for ipv6.";
            }
            leaf ipv6-group-address {
              type inet:ipv6-address;
              description "Group address in I-PMSI or S-PMSI for ipv6.";
            }
          }
        }
      }

      grouping mvpn-ipmsi-tunnel-state-ipv4 {
        description
          "Default mdt or I-PMSI operational state information";
        container mvpn-ipmsi-tunnel-info {
          description
            "Default mdt or I-PMSI operational state information";
```

```
          uses mvpn-pmsi-state;
          uses mvpn-pmsi-entry-ipv4;
        }
      }

      grouping mvpn-ipmsi-tunnel-state-ipv6 {
        description
          "Default mdt or I-PMSI operational state information";
        container mvpn-ipmsi-tunnel-info {
          description
            "Default mdt or I-PMSI operational state information";
          uses mvpn-pmsi-state;
          uses mvpn-pmsi-entry-ipv6;
        }
      }

      grouping mvpn-spmsi-tunnel-state-ipv4 {
        description
          "Data mdt or S-PMSI operational state information";
        container mvpn-spmsi-tunnel-ipv4-infos {
          description
            "Data mdt or S-PMSI operational state information";
          list mvpn-spmsi-tunnel-ipv4-info {
            key "tunnel-mode";
            description
              "Data mdt or S-PMSI operational state information";
            uses mvpn-pmsi-state;
            uses mvpn-pmsi-entry-ipv4;
          }
        }
      }

      grouping mvpn-spmsi-tunnel-state-ipv6 {
        description
          "Data mdt or S-PMSI operational state information";
        container mvpn-spmsi-tunnel-ipv6-infos {
          description
            "Data mdt or S-PMSI operational state information";
          list mvpn-spmsi-tunnel-ipv6-info {
            key "tunnel-mode";
            description
              "Data mdt or S-PMSI operational state information";
            uses mvpn-pmsi-state;
            uses mvpn-pmsi-entry-ipv6;
          }
        }
      }

      grouping l3vpn-mvrf-params {
```

```
        description
          "Specify multicast vrf parameters and provide
           multicast vrf operational state information";
        container ipv4 {
          description
            "Specify multicast ipv4 vrf parameters and provide
             multicast ipv4 vrf operational state information";
            container mvpn {
              description "Specify multicast ipv4 vrf parameters";
              uses mvpn-instance-config;
              uses mvpn-vpn-targets;
              uses mvpn-ipmsi-tunnel-config;
              uses mvpn-spmsi-tunnel-config-ipv4;
            }
            container mvpn-state {
              config "false";
              description
                "Multicast ipv4 vrf operational state information";
              uses mvpn-instance-config;
              uses mvpn-vpn-targets;
              uses mvpn-ipmsi-tunnel-config;
              uses mvpn-spmsi-tunnel-config-ipv4;
              uses mvpn-ipmsi-tunnel-state-ipv4;
              uses mvpn-spmsi-tunnel-state-ipv4;
            }
          }
        container ipv6 {
          description
            "Ipv6 address family specific multicast vrf parameters and
             multicast vrf operational state information";
            container mvpn {
              description "Ipv6 address family specific multicast vrf parameters
";
              uses mvpn-instance-config;
              uses mvpn-vpn-targets;
              uses mvpn-ipmsi-tunnel-config;
              uses mvpn-spmsi-tunnel-config-ipv6;
            }
            container mvpn-state {
              config "false";
              description
                "Ipv6 address family multicast vrf operational state information
";
              uses mvpn-instance-config;
              uses mvpn-vpn-targets;
              uses mvpn-ipmsi-tunnel-config;
              uses mvpn-spmsi-tunnel-config-ipv6;
              uses mvpn-ipmsi-tunnel-state-ipv6;
              uses mvpn-spmsi-tunnel-state-ipv6;
            }
        }
```

```
      }


      augment "/ni:network-instances/ni:network-instance" {
        description
          "Augment network instance container for per multicast VRF configuratio
n";
        container l3vpn {
          description
            "Configuration of multicast vpn specific parameters and
             operational state of multicast vpn specific parameters";
          uses l3vpn-mvrf-params;
        }
      }

   }
   <CODE ENDS>
```

5. Security Considerations

The data model defined does not introduce any security implications. This draft does not change any underlying security issues inherent in [RFC8022].

6. IANA Considerations

TBD

7. References

7.1. Normative References

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-05 (work in progress), October 2015.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, November 2016.

7.2. Informative References

[RFC6037] Rosen, E., Cai, Y., and IJ. Wijnands, "Cisco Systems' Solution for Multicast in BGP/MPLS IP VPNs", RFC 6037, October 2010.

[RFC6513] Rosen, E. and R. Aggarwal, "Multicast in MPLS/BGP IP VPNs", RFC 6513, February 2012.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, February 2012.

[RFC7246] IJ. Wijnands, P. Hitchen, N. Leymann, W. Henderickx, A. Gulko and J. Tantsura, " Multipoint Label Distribution Protocol In-Band Signaling in a Virtual Routing and Forwarding (VRF) Table Context ", RFC 7246, June 2014.

[RFC7900] Y. Rekhter, E. Rosen, R. Aggarwal, Arktan, Y. Cai and T. Morin, " Extranet Multicast in BGP/IP MPLS VPNs ", RFC 7900, June 2016.

8. Acknowledgments

The authors would like to thank Anish Peter, Stig Venaas for their valuable contributions.

Authors' Addresses

Yisong Liu
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: liuyisong@huawei.com

Feng Guo
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: guofeng@huawei.com

Xufeng Liu
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

Email: Xufeng_Liu@jabil.com

Robert Kebler
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: rkebler@juniper.net

Mahesh Sivakumar
Cisco Systems, Inc
510 McCarthy Blvd
Milpitas, California 95035
USA

Email: masivaku@cisco.com

BESS Working Group
Internet Draft
Intended status: Standards Track
Expires: January 3, 2018

Y. Liu
F. Guo
Huawei Technologies
X. Liu
Jabil
R. Kebler
Juniper Networks
M. Sivakumar
Cisco

July 3, 2017

Yang Data Model for Multicast in MPLS/BGP IP VPNs
draft-liu-bess-mvpn-yang-04

Status of this Memo

Copyright Notice

Abstract

This document defines a YANG data model that can be used to configure and manage multicast in MPLS/BGP IP VPNs.

Table of Contents

1. Introduction 2
   1.1. Requirements Language..... 3
   1.2. Terminology 3
2. Design of Data model..... 3
   2.1. Scope of model 3
   2.2. Optional capabilities..... 3
   2.3. Position of address family in hierarchy..... 4
3. Module Structure 4
   3.1. MVPN Configuration..... 4
   3.2. MVPN Operational State..... 7
4. MVPN YANG Modules 12
5. Security Considerations..... 30
6. IANA Considerations 30
7. References 30
   7.1. Normative References..... 30
   7.2. Informative References..... 31
8. Acknowledgments 31

1. Introduction

YANG [RFC6020] is a data definition language that was introduced to define the contents of a conceptual data store that allows networked devices to be managed using NETCONF [RFC6241]. YANG is proving relevant beyond its initial confines, as bindings to other interfaces (e.g. REST) and encoding other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interface, such as CLI and Programmatic APIs.

This document defines a YANG data model that can be used to configure and manage Multicast in MPLS/BGP IP VPN (MVPN). It includes Cisco systems' solution [RFC6037], BGP MVPN [RFC6513] [RFC6514] etc. Currently this model is incomplete, but it will support the core MVPN protocols, as well as many other features mentioned in separate MVPN RFCs. In addition, Non-core features described in MVPN standards other than mentioned above RFC in future version.

1.1. 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 RFC-2119 [RFC2119].

1.2. Terminology

The terminology for describing YANG data models is found in [RFC6020].

This draft employs YANG tree diagrams, which are explained in [I-D.ietf-netmod-rfc6087bis].

2. Design of Data model

2.1. Scope of model

The model covers Rosen MVPN [RFC6037], BGP MVPN [RFC6513] [RFC6514]. The representation of some of extension features is not completely specified in this draft of the data model. This model is being circulated in its current form for early oversight and review of the basic hierarchy.

The operational state fields of this model are also incomplete, though the structure of what has been written may be taken as representative of the structure of the model when complete.

This model does not cover other MVPN related protocols such as MVPN Extranet [RFC7900] or MVPN MLDP In-band signaling [RFC7246] etc., these will be covered by future Internet Drafts.

2.2. Optional capabilities

This model is designed to represent the capabilities of MVPN devices with various specifications, including some with basic subsets of the MVPN protocols. The main design goals of this draft are that any major now-existing implementation may be said to support the basic model, and that the configuration of all implementations meeting the specification is easy to express through some combination of the features in the basic model and simple vendor augmentations.

On the other hand, operational state parameters are not so widely designated as features, as there are many cases where the defaulting of an operational state parameter would not cause any harm to the system, and it is much more likely that an implementation without native support for a piece of operational state would be able to

derive a suitable value for a state variable that is not natively supported.

For the same reason, wide constant ranges (for example, timer maximum and minimum) will be used in the model. It is expected that vendors will augment the model with any specific restrictions that might be required. Vendors may also extend the features list with proprietary extensions.

2.3. Position of address family in hierarchy

The current draft contains MVPN IPv4 and IPv6 as separate schema branches in the structure. The reason for this is to inherit l3vpn yang model structure and make it easier for implementations which may optionally choose to support specific address families. And the names of objects may be different between the IPv4 and IPv6 address families.

3. Module Structure

3.1. MVPN Configuration

The MVPN modules define the network-instance-wide configuration options in a two-level hierarchy as listed below:

Instance level: MVPN configuration attributes for the entire routing instance, including route-target, I-PMSI tunnel and S-PMSI number, common timer etc.

PMSI tunnel level: MVPN configuration attributes applicable to the I-PMSI and per S-PMSI tunnel configuration attributes, including tunnel mode, tunnel specific parameters and threshold etc.

Where fields are not genuinely essential to protocol operation, they are marked as optional. Some fields will be essential but have a default specified, so that they need not be configured explicitly. The module structure also applies, where applicable, to the operational state as well.

Our current direction is to agree to a network-instance-centric (VRF) model as opposed to protocol-centric mainly because it inherits l3vpn and it can as a part of l3vpn yang model.

The MVPN model will augment "/ni:network-instances/ni:network-instance/l3vpn:".

```
  augment /ni:network-instances/ni:network-instance:
    +--rw l3vpn
       +--rw ipv4
```

```
|  +--rw mvpn
|  |  +--rw signaling-mode?        enumeration
|  |  +--rw auto-discovery-mode?   enumeration
|  |  +--rw config-type?           enumeration
|  |  +--rw is-sender-site?        boolean
|  |  +--rw rpt-spt-mode?          enumeration
|  |  +--rw mvpn-route-targets
|  |  |  +--rw mvpn-route-target* [rt-type rt-value]
|  |  |     +--rw rt-type     enumeration
|  |  |     +--rw rt-value    string
|  |  +--rw mvpn-ipmsi-tunnel
|  |  |  +--rw tunnel-type?                enumeration
|  |  |  +--rw (ipmsi-tunnel-attribute)?
|  |  |     +--:(p2mp-te)
|  |  |     |  +--rw te-p2mp-template?           string
|  |  |     +--:(p2mp-mldp)
|  |  |     +--:(pim-ssm)
|  |  |     |  +--rw ssm-default-group-addr?     inet:ip-address
|  |  |     +--:(pim-sm)
|  |  |     |  +--rw sm-default-group-addr?      inet:ip-address
|  |  |     +--:(bidir-pim)
|  |  |     |  +--rw bidir-default-group-addr?   inet:ip-address
|  |  |     +--:(ingress-replication)
|  |  |     +--:(mp2mp-mldp)
|  |  +--rw mvpn-spmsi-tunnels
|  |     +--rw switch-delay-time?           uint8
|  |     +--rw switch-back-holddown-time?   uint16
|  |     +--rw tunnel-limit?                uint16
|  |     +--rw mvpn-spmsi-tunnel* [tunnel-type]
|  |        +--rw tunnel-type                    enumeration
|  |        +--rw (spmsi-tunnel-attribute)?
|  |        |  +--:(p2mp-te)
|  |        |  |  +--rw te-p2mp-template?              string
|  |        |  +--:(p2mp-mldp)
|  |        |  +--:(pim-ssm)
|  |        |  |  +--rw ssm-group-pool-addr?           inet:ip-address
|  |        |  |  +--rw ssm-group-pool-masklength?     uint8
|  |        |  +--:(pim-sm)
|  |        |  |  +--rw sm-group-pool-addr?            inet:ip-address
|  |        |  |  +--rw sm-group-pool-masklength?      uint8
|  |        |  +--:(bidir-pim)
|  |        |  |  +--rw bidir-group-pool-addr?         inet:ip-address
|  |        |  |  +--rw bidir-group-pool-masklength?   uint8
```

```
          |  |        |  +--:(ingress-replication)
          |  |        |  +--:(mp2mp-mldp)
          |  |        +--rw switch-threshold?              uint32
          |  |        +--rw switch-wildcard-mode?          enumeration
          |  |        +--rw (address-mask-or-acl)?
          |  |           +--:(address-mask)
          |  |           |  +--rw ipv4-group-addr?              inet:ipv4-addre
ss
          |  |           |  +--rw ipv4-group-masklength?        uint8
          |  |           |  +--rw ipv4-source-addr?             inet:ipv4-addre
ss
          |  |           |  +--rw ipv4-source-masklength?       uint8
          |  |           +--:(acl)
          |  |              +--rw group-acl-ipv4?               string

     augment /ni:network-instances/ni:network-instance:
       +--rw l3vpn
          +--rw ipv6
             +--rw mvpn
                |  +--rw signaling-mode?        enumeration
                |  +--rw auto-discovery-mode?   enumeration
                |  +--rw config-type?           enumeration
                |  +--rw is-sender-site?        boolean
                |  +--rw rpt-spt-mode?          enumeration
                |  +--rw mvpn-route-targets
                |  |  +--rw mvpn-route-target* [rt-type rt-value]
                |  |     +--rw rt-type     enumeration
                |  |     +--rw rt-value    string
                |  +--rw mvpn-ipmsi-tunnel
                |  |  +--rw tunnel-type?                enumeration
                |  |  +--rw (ipmsi-tunnel-attribute)?
                |  |     +--:(p2mp-te)
                |  |     |  +--rw te-p2mp-template?           string
                |  |     +--:(p2mp-mldp)
                |  |     +--:(pim-ssm)
                |  |     |  +--rw ssm-default-group-addr?     inet:ip-address
                |  |     +--:(pim-sm)
                |  |     |  +--rw sm-default-group-addr?      inet:ip-address
                |  |     +--:(bidir-pim)
                |  |     |  +--rw bidir-default-group-addr?   inet:ip-address
                |  |     +--:(ingress-replication)
                |  |     +--:(mp2mp-mldp)
                |  +--rw mvpn-spmsi-tunnels
                |     +--rw switch-delay-time?           uint8
                |     +--rw switch-back-holddown-time?   uint16
                |     +--rw tunnel-limit?                uint16
                |     +--rw mvpn-spmsi-tunnel* [tunnel-type]
```

```
               |        +--rw tunnel-type                    enumeration
               |        +--rw (spmsi-tunnel-attribute)?
               |        |  +--:(p2mp-te)
               |        |  |  +--rw te-p2mp-template?          string
               |        |  +--:(p2mp-mldp)
               |        |  +--:(pim-ssm)
               |        |  |  +--rw ssm-group-pool-addr?       inet:ip-address
               |        |  |  +--rw ssm-group-pool-masklength? uint8
               |        |  +--:(pim-sm)
               |        |  |  +--rw sm-group-pool-addr?        inet:ip-address
               |        |  |  +--rw sm-group-pool-masklength?  uint8
               |        |  +--:(bidir-pim)
               |        |  |  +--rw bidir-group-pool-addr?     inet:ip-address
               |        |  |  +--rw bidir-group-pool-masklength? uint8
               |        |  +--:(ingress-replication)
               |        |  +--:(mp2mp-mldp)
               |        +--rw switch-threshold?              uint32
               |        +--rw switch-wildcard-mode?          enumeration
               |        +--rw (address-mask-or-acl)?
               |           +--:(address-mask)
               |           |  +--rw ipv6-group-addr?           inet:ipv6-addre
ss
               |           |  +--rw ipv6-groupmasklength?      uint8
               |           |  +--rw ipv6-source-addr?          inet:ipv6-addre
ss
               |           |  +--rw ipv6-source-masklength?    uint8
               |           +--:(acl)
               |              +--rw group-acl-ipv6?            String
```

3.2. MVPN Operational State

The MVPN module contains operational state information also in a two-level hierarchy as mentioned earlier.

Instance level: MVPN operational state attributes for the entire routing instance, including route-target, I-PMSI tunnel and S-PMSI number, common timer etc.

PMSI tunnel level: MVPN PMSI tunnel operational state attributes applicable to the I-PMSI and per S-PMSI tunnel operational state attributes, including tunnel mode, tunnel role, tunnel specific parameters and referenced private source and group address etc.

```
  augment /ni:network-instances/ni:network-instance:
    +--rw l3vpn
       +--rw ipv4
```

```
|  +--rw mvpn
|  |--...
|  +--ro mvpn-state
|     +--ro signaling-mode?                enumeration
|     +--ro auto-discovery-mode?           enumeration
|     +--ro config-type?                   enumeration
|     +--ro is-sender-site?                boolean
|     +--ro rpt-spt-mode?                  enumeration
|     +--ro mvpn-route-targets
|     |  +--ro mvpn-route-target* [rt-type rt-value]
|     |     +--ro rt-type     enumeration
|     |     +--ro rt-value    string
|     +--ro mvpn-ipmsi-tunnel
|     |  +--ro tunnel-type?               enumeration
|     |  +--ro (ipmsi-tunnel-attribute)?
|     |     +--:(p2mp-te)
|     |     |  +--ro te-p2mp-template?          string
|     |     +--:(p2mp-mldp)
|     |     +--:(pim-ssm)
|     |     |  +--ro ssm-default-group-addr?    inet:ip-address
|     |     +--:(pim-sm)
|     |     |  +--ro sm-default-group-addr?     inet:ip-address
|     |     +--:(bidir-pim)
|     |     |  +--ro bidir-default-group-addr?  inet:ip-address
|     |     +--:(ingress-replication)
|     |     +--:(mp2mp-mldp)
|     +--ro mvpn-spmsi-tunnels
|     |  +--ro switch-delay-time?          uint8
|     |  +--ro switch-back-holddown-time?  uint16
|     |  +--ro tunnel-limit?               uint16
|     |  +--ro mvpn-spmsi-tunnel* [tunnel-type]
|     |     +--ro tunnel-type                    enumeration
|     |     +--ro (spmsi-tunnel-attribute)?
|     |     |  +--:(p2mp-te)
|     |     |  |  +--ro te-p2mp-template?             string
|     |     |  +--:(p2mp-mldp)
|     |     |  +--:(pim-ssm)
|     |     |  |  +--ro ssm-group-pool-addr?          inet:ip-address
|     |     |  |  +--ro ssm-group-pool-masklength?    uint8
|     |     |  +--:(pim-sm)
|     |     |  |  +--ro sm-group-pool-addr?           inet:ip-address
|     |     |  |  +--ro sm-group-pool-masklength?     uint8
|     |     |  +--:(bidir-pim)
|     |     |  |  +--ro bidir-group-pool-addr?        inet:ip-address
|     |     |  |  +--ro bidir-group-pool-masklength?  uint8
|     |     |  +--:(ingress-replication)
|     |     |  +--:(mp2mp-mldp)
|     |     +--ro switch-threshold?              uint32
|     |     +--ro switch-wildcard-mode?          enumeration
```

```
          |     |     +--ro (address-mask-or-acl)?
          |     |        +--:(address-mask)
          |     |        |  +--ro ipv4-group-addr?              inet:ipv4-addre
ss
          |     |        |  +--ro ipv4-group-masklength?        uint8
          |     |        |  +--ro ipv4-source-addr?             inet:ipv4-addre
ss
          |     |        |  +--ro ipv4-source-masklength?       uint8
          |     |        +--:(acl)
          |     |           +--ro group-acl-ipv4?               string
          |     +--ro mvpn-ipmsi-tunnel-info
          |     |  +--ro tunnel-type?             enumeration
          |     |  +--ro (pmsi-tunnel-attribute)?
          |     |  |  +--:(p2mp-te)
          |     |  |  |  +--ro te-p2mp-id?              uint16
          |     |  |  |  +--ro te-tunnel-id?            uint16
          |     |  |  |  +--ro te-extend-tunnel-id?     uint16
          |     |  |  +--:(p2mp-mldp)
          |     |  |  |  +--ro mldp-root-addr?          inet:ip-address
          |     |  |  |  +--ro mldp-lsp-id?             string
          |     |  |  +--:(pim-ssm)
          |     |  |  |  +--ro ssm-group-addr?          inet:ip-address
          |     |  |  +--:(pim-sm)
          |     |  |  |  +--ro sm-group-addr?           inet:ip-address
          |     |  |  +--:(bidir-pim)
          |     |  |  |  +--ro bidir-group-addr?        inet:ip-address
          |     |  |  +--:(ingress-replication)
          |     |  |  +--:(mp2mp-mldp)
          |     |  +--ro tunnel-role?             enumeration
          |     |  +--ro mvpn-pmsi-sg-ref-ipv4s
          |     |     +--ro mvpn-pmsi-sg-ref-ipv4* [ipv4-source-address ipv4-gro
up-address]
          |     |        +--ro ipv4-source-address    inet:ipv4-address
          |     |        +--ro ipv4-group-address     inet:ipv4-address
          |     +--ro mvpn-spmsi-tunnel-ipv4-infos
          |        +--ro mvpn-spmsi-tunnel-ipv4-info* [tunnel-type]
          |           +--ro tunnel-type               enumeration
          |           +--ro (pmsi-tunnel-attribute)?
          |           |  +--:(p2mp-te)
          |           |  |  +--ro te-p2mp-id?              uint16
          |           |  |  +--ro te-tunnel-id?            uint16
          |           |  |  +--ro te-extend-tunnel-id?     uint16
          |           |  +--:(p2mp-mldp)
          |           |  |  +--ro mldp-root-addr?          inet:ip-address
          |           |  |  +--ro mldp-lsp-id?             string
          |           |  +--:(pim-ssm)
          |           |  |  +--ro ssm-group-addr?          inet:ip-address
          |           |  +--:(pim-sm)
          |           |  |  +--ro sm-group-addr?           inet:ip-address
          |           |  +--:(bidir-pim)
          |           |  |  +--ro bidir-group-addr?        inet:ip-address
          |           |  +--:(ingress-replication)
```

```
          |           |  +--:(mp2mp-mldp)
          |           +--ro tunnel-role?              enumeration
          |           +--ro mvpn-pmsi-sg-ref-ipv4s
          |              +--ro mvpn-pmsi-sg-ref-ipv4* [ipv4-source-address ipv4-
group-address]
          |                 +--ro ipv4-source-address    inet:ipv4-address
          |                 +--ro ipv4-group-address     inet:ipv4-address
          +--rw ipv6
             +--rw mvpn
             |--...
             +--ro mvpn-state
                +--ro signaling-mode?                 enumeration
                +--ro auto-discovery-mode?            enumeration
                +--ro config-type?                    enumeration
                +--ro is-sender-site?                 boolean
                +--ro rpt-spt-mode?                   enumeration
                +--ro mvpn-route-targets
                |  +--ro mvpn-route-target* [rt-type rt-value]
                |     +--ro rt-type     enumeration
                |     +--ro rt-value    string
                +--ro mvpn-ipmsi-tunnel
                |  +--ro tunnel-type?                enumeration
                |  +--ro (ipmsi-tunnel-attribute)?
                |     +--:(p2mp-te)
                |     |  +--ro te-p2mp-template?           string
                |     +--:(p2mp-mldp)
                |     +--:(pim-ssm)
                |     |  +--ro ssm-default-group-addr?     inet:ip-address
                |     +--:(pim-sm)
                |     |  +--ro sm-default-group-addr?      inet:ip-address
                |     +--:(bidir-pim)
                |     |  +--ro bidir-default-group-addr?   inet:ip-address
                |     +--:(ingress-replication)
                |     +--:(mp2mp-mldp)
                +--ro mvpn-spmsi-tunnels
                |  +--ro switch-delay-time?           uint8
                |  +--ro switch-back-holddown-time?   uint16
                |  +--ro tunnel-limit?                uint16
                |  +--ro mvpn-spmsi-tunnel* [tunnel-type]
                |     +--ro tunnel-type                      enumeration
                |     +--ro (spmsi-tunnel-attribute)?
                |     |  +--:(p2mp-te)
                |     |  |  +--ro te-p2mp-template?              string
                |     |  +--:(p2mp-mldp)
                |     |  +--:(pim-ssm)
                |     |  |  +--ro ssm-group-pool-addr?           inet:ip-address
                |     |  |  +--ro ssm-group-pool-masklength?     uint8
                |     |  +--:(pim-sm)
                |     |  |  +--ro sm-group-pool-addr?            inet:ip-address
                |     |  |  +--ro sm-group-pool-masklength?      uint8
```

```
                |     |  +--:(bidir-pim)
                |     |  |  +--ro bidir-group-pool-addr?         inet:ip-address
                |     |  |  +--ro bidir-group-pool-masklength?   uint8
                |     |  +--:(ingress-replication)
                |     |  +--:(mp2mp-mldp)
                |     +--ro switch-threshold?              uint32
                |     +--ro switch-wildcard-mode?          enumeration
                |     +--ro (address-mask-or-acl)?
                |        +--:(address-mask)
                |        |  +--ro ipv6-group-addr?               inet:ipv6-addre
ss
                |        |  +--ro ipv6-groupmasklength?          uint8
                |        |  +--ro ipv6-source-addr?              inet:ipv6-addre
ss
                |        |  +--ro ipv6-source-masklength?        uint8
                |        +--:(acl)
                |           +--ro group-acl-ipv6?                string
                +--ro mvpn-ipmsi-tunnel-info
                |  +--ro tunnel-type?              enumeration
                |  +--ro (pmsi-tunnel-attribute)?
                |  |  +--:(p2mp-te)
                |  |  |  +--ro te-p2mp-id?              uint16
                |  |  |  +--ro te-tunnel-id?            uint16
                |  |  |  +--ro te-extend-tunnel-id?     uint16
                |  |  +--:(p2mp-mldp)
                |  |  |  +--ro mldp-root-addr?          inet:ip-address
                |  |  |  +--ro mldp-lsp-id?             string
                |  |  +--:(pim-ssm)
                |  |  |  +--ro ssm-group-addr?          inet:ip-address
                |  |  +--:(pim-sm)
                |  |  |  +--ro sm-group-addr?           inet:ip-address
                |  |  +--:(bidir-pim)
                |  |  |  +--ro bidir-group-addr?        inet:ip-address
                |  |  +--:(ingress-replication)
                |  |  +--:(mp2mp-mldp)
                |  +--ro tunnel-role?              enumeration
                |  +--ro mvpn-pmsi-sg-ref-ipv6s
                |     +--ro mvpn-pmsi-sg-ref-ipv6* [ipv6-source-address ipv6-gro
up-address]
                |        +--ro ipv6-source-address    inet:ipv6-address
                |        +--ro ipv6-group-address     inet:ipv6-address
                +--ro mvpn-spmsi-tunnel-ipv6-infos
                   +--ro mvpn-spmsi-tunnel-ipv6-info* [tunnel-type]
                      +--ro tunnel-type               enumeration
                      +--ro (pmsi-tunnel-attribute)?
                      |  +--:(p2mp-te)
                      |  |  +--ro te-p2mp-id?              uint16
                      |  |  +--ro te-tunnel-id?            uint16
                      |  |  +--ro te-extend-tunnel-id?     uint16
                      |  +--:(p2mp-mldp)
                      |  |  +--ro mldp-root-addr?          inet:ip-address
                      |  |  +--ro mldp-lsp-id?             string
```

```
                      |  +--:(pim-ssm)
                      |  |  +--ro ssm-group-addr?          inet:ip-address
                      |  +--:(pim-sm)
                      |  |  +--ro sm-group-addr?           inet:ip-address
                      |  +--:(bidir-pim)
                      |  |  +--ro bidir-group-addr?        inet:ip-address
                      |  +--:(ingress-replication)
                      |  +--:(mp2mp-mldp)
                      +--ro tunnel-role?              enumeration
                      +--ro mvpn-pmsi-sg-ref-ipv6s
                         +--ro mvpn-pmsi-sg-ref-ipv6* [ipv6-source-address ipv6-
group-address]
                            +--ro ipv6-source-address    inet:ipv6-address
                            +--ro ipv6-group-address     inet:ipv6-address
```

4. MVPN YANG Modules

```
   <CODE BEGINS> file "ietf-mvpn@2017-07-03.yang"
   module ietf-mvpn {
      namespace "urn:ietf:params:xml:ns:yang:ietf-mvpn";
      prefix mvpn;

      import ietf-network-instance {
       prefix ni;
      }

      import ietf-inet-types {
        prefix inet;
      }


      organization
        "IETF BESS(BGP Enabled Services) Working Group";
      contact
        "liuyisong@huawei.com
         guofeng@huawei.com
         xliu@kuatrotech.com
         rkebler@juniper.net
         masivaku@cisco.com";
      description
        "This YANG module defines the generic configuration
         data for mvpn, which is common across all of the vendor
         implementations of the protocol. It is intended that the module
         will be extended by vendors to define vendor-specific
         mvpn configuration parameters.";

      revision 2017-07-03 {
        description
          "Update S-PMSI configuration and errata.";
```

```
        reference
          "RFC XXXX: A YANG Data Model for MVPN";
      }

      revision 2016-10-28 {
        description
          "Initial revision.";
        reference
          "RFC XXXX: A YANG Data Model for MVPN";
      }

      grouping mvpn-instance-config {
        description "Mvpn basic configuration per instance.";

        leaf signaling-mode {
          type enumeration {
            enum invalid {
              value "0";
              description "invalid";
            }
            enum bgp {
              value "1";
              description "bgp";
            }
            enum pim {
              value "2";
              description "pim";
            }
            enum mldp {
              value "3";
              description "mldp";
            }
          }
          default "invalid";
          description "Signaling mode for C-multicast route.";
        }
        leaf auto-discovery-mode {
          type enumeration {
            enum none {
              value "0";
              description "none";
            }
            enum ad {
              value "1";
              description "auto-discovery by BGP";
            }
          }
          default "none";
          description "Auto discovery mode.";
```

```
        }
        leaf config-type {
          type enumeration {
            enum md {
              value "0";
              description "md(rosen)";
            }
            enum ng {
              value "1";
              description "ng";
            }
          }
          default "md";
          description "Mvpn type, which can be md(rosen) mvpn or ng mvpn.";
        }
        leaf is-sender-site {
          type boolean;
          default "false";
          description "Configure the current PE as a sender PE.";
        }
        leaf rpt-spt-mode {
          type enumeration {
            enum spt-only {
              value "0";
              description
                "Only spt mode for crossing public net.";
            }
            enum rpt-spt {
              value "1";
              description
                "Both rpt and spt mode for corssing public net.";
            }
          }
          default "spt-only";
          description
            "ASM mode in multicast private net for crossing public net.";
        }

      }

      grouping mvpn-vpn-targets {
        description "May be different from l3vpn unicast route-targets";
        container mvpn-route-targets{
          description "Multicast vpn route-targets";
          list mvpn-route-target {
            key "rt-type rt-value" ;
            description
              "List of multicast route-targets" ;
            leaf rt-type {
```

```
            type enumeration {
              enum export-extcommunity {
                value "0";
                description "export-extcommunity";
              }
              enum import-extcommunity {
                value "1";
                description "import-extcommunity";
              }
            }
            mandatory "true";
            description
              "rt types are as follows:
               export-extcommunity: specifies the value of
               the extended community attribute of the
               route from an outbound interface to the
               destination vpn.
               import-extcommunity: receives routes that
               carry the specified extended community
               attribute";
          }
          leaf rt-value {
            type string {
              length "3..21";
            }
            description
              "the available mvpn target formats are as
               follows:
               - 16-bit as number:32-bit user-defined
               number, for example, 1:3. an as number
               ranges from 0 to 65535, and a user-defined
               number ranges from 0 to 4294967295. The as
               number and user-defined number cannot be
               both 0s. That is, a vpn target cannot be 0:0.
               - 32-bit ip address:16-bit user-defined
               number, for example, 192.168.122.15:1.
               The ip address ranges from 0.0.0.0 to
               255.255.255.255, and the user-defined
               number ranges from 0 to 65535.";
          }
        }
      }
    }


  grouping mvpn-ipmsi-tunnel-config {
    description "Default mdt for rosen mvpn and I-PMSI for ng mvpn";
```

```
        container mvpn-ipmsi-tunnel {
          description "I-PMSI tunnel configuraton";
          leaf tunnel-type {
            type enumeration {
              enum invalid {
                value "0";
                description "invalid";
              }
              enum p2mp-te {
                value "1";
                description "p2mp-te";
              }
              enum p2mp-mldp {
                value "2";
                description "p2mp-mldp";
              }
              enum pim-ssm {
                value "3";
                description "pim-ssm";
              }
              enum pim-sm {
                value "4";
                description "pim-sm";
              }
              enum bidir-pim {
                value "5";
                description "bidir-pim";
              }
              enum ingress-replication {
                value "6";
                description "ingress-replication";
              }
              enum mp2mp-mldp {
                value "7";
                description "mp2mp-mldp";
              }
            }
            description "I-PMSI tunnel type.";
          }
          choice ipmsi-tunnel-attribute {
            description "I-PMSI tunnel attributes configuration";
            case p2mp-te {
              description "P2mp TE tunnel";
              leaf te-p2mp-template {
                type string {
                  length "1..31";
                }
                description "P2mp te tunnel template";
              }
```

```
            }
            case p2mp-mldp {
              description "Mldp tunnel";
            }
            case pim-ssm {
              description "Pim ssm tunnel";
              leaf ssm-default-group-addr {
                type inet:ip-address;
                description "Default mdt or I-PMSI group address.";
              }
            }
            case pim-sm {
              description "Pim sm tunnel";
              leaf sm-default-group-addr {
                type inet:ip-address;
                description "Default mdt or I-PMSI group address.";
              }
            }
            case bidir-pim {
              description "Bidir pim tunnel";
              leaf bidir-default-group-addr {
                type inet:ip-address;
                description "Default mdt or I-PMSI group address.";
              }
            }
            case ingress-replication {
              description "Ingress replication p2p tunnel";
            }
            case mp2mp-mldp {
              description "Mp2mp mldp tunnel";
            }
          }
        }
      }

      grouping mvpn-spmsi-tunnel-basic-config {
        description "S-PMSI tunnel basic configuration";
        leaf tunnel-type {
          type enumeration {
            enum invalid {
              value "0";
              description "invalid";
            }
            enum p2mp-te {
              value "1";
              description "p2mp-te";
            }
            enum p2mp-mldp {
              value "2";
```

```
            description "p2mp-mldp";
          }
          enum pim-ssm {
            value "3";
            description "pim-ssm";
          }
          enum pim-sm {
            value "4";
            description "pim-sm";
          }
          enum bidir-pim {
            value "5";
            description "bidir-pim";
          }
          enum ingress-replication {
            value "6";
            description "ingress-replication";
          }
          enum mp2mp-mldp {
            value "7";
            description "mp2mp-mldp";
          }
        }
        description "S-PMSI tunnel type.";
      }
      choice spmsi-tunnel-attribute {
        description "S-PMSI tunnel attributes configuration";
        case p2mp-te {
          description "P2mp te tunnel";
          leaf te-p2mp-template {
            type string {
              length "1..31";
            }
            description "P2mp te tunnel template";
          }
        }
        case p2mp-mldp {
          description "Mldp tunnel";
        }
        case pim-ssm {
          description "Pim ssm tunnel";
          leaf ssm-group-pool-addr {
            type inet:ip-address;
            description "Group pool address for data mdt or pim s-pmsi.";
          }
          leaf ssm-group-pool-masklength {
            type uint8 {
              range "8..128";
            }
```

```
            description "Group pool mask for data mdt or pim s-pmsi";
          }
        }
        case pim-sm {
          description "Pim sm tunnel";
          leaf sm-group-pool-addr {
            type inet:ip-address;
            description "Group pool address for data mdt or pim s-pmsi.";
          }
          leaf sm-group-pool-masklength {
            type uint8 {
              range "8..128";
            }
            description "Group pool mask for data mdt or pim s-pmsi";
          }
        }
        case bidir-pim {
          description "Bidir pim tunnel";
          leaf bidir-group-pool-addr {
            type inet:ip-address;
            description "Group pool address for data mdt or pim s-pmsi.";
          }
          leaf bidir-group-pool-masklength {
            type uint8 {
              range "8..128";
            }
            description "Group pool mask for data mdt or pim s-pmsi";
          }
        }
        case ingress-replication {
          description "Ingress replication p2p tunnel";
        }
        case mp2mp-mldp {
          description "Mp2mp mldp tunnel";
        }
      }
      leaf switch-threshold {
        type uint32 {
          range "0..4194304";
        }
        default "0";
        description
          "Multicast packet rate threshold for
           triggering the switching from the
           I-PMSI to the S-PMSI. The value is
           an integer ranging from 0 to 4194304, in
           kbit/s. The default value is 0.";
      }
      leaf switch-wildcard-mode {
```

```
          type enumeration {
            enum source-group {
              value "0";
              description
                "Wildcard neither for source or group address.";
            }
            enum star-star {
              value "1";
              description
                "Wildcard for both source and group address.";
            }
            enum star-group {
              value "2";
              description
                "Wildcard only for source address.";
            }
            enum source-star {
              value "3";
              description
                "Wildcard only for group address.";
            }
          }
          default "source-group";
          description
            "I-PMSI switching to S-PMSI mode for private net
            wildcard mode, which including (*,*), (*,G), (S,*),
            (S,G) four modes.";
        }
      }

      grouping mvpn-spmsi-tunnel-config-ipv4 {
        description "Data mdt for rosen mvpn and S-PMSI for ng mvpn";

        container mvpn-spmsi-tunnels {
          description "S-PMSI tunnel configuration";
          leaf switch-delay-time {
            type uint8 {
              range "3..60";
            }
            units seconds;
            default "5";
            description
            "Delay for switching from the I-PMSI to
             the S-PMSI. The value is an integer
             ranging from 3 to 60, in seconds. ";
          }
          leaf switch-back-holddown-time {
            type uint16 {
              range "0..512";
```

```
            }
            units seconds;
            default "60";
            description
              "Delay for switching back from the S-PMSI
               to the I-PMSI. The value is an integer
               ranging from 0 to 512, in seconds. ";
          }
          leaf tunnel-limit {
            type uint16 {
              range "1..1024";
            }
            description
              "Maximum number of s-pmsi tunnels allowed.";
          }

          list mvpn-spmsi-tunnel {
            key "tunnel-type";
            description "S-PMSI tunnel attributes configuration";

            uses mvpn-spmsi-tunnel-basic-config;

            choice address-mask-or-acl {
              description "Type of define private net multicast address range";
              case address-mask {
                description "Use the type of address and mask";
                leaf ipv4-group-addr {
                  type inet:ipv4-address;
                  description
                    "Start and end ipv4 addresses of the group
                     address in private net. ";
                }
                leaf ipv4-group-masklength {
                  type uint8 {
                    range "4..32";
                  }
                  description
                    "Group mask length for ipv4 addresses in
                     the group address pool in private net.";
                }
                leaf ipv4-source-addr {
                  type inet:ipv4-address;
                  description
                    "Start and end ipv4 addresses of the source
                     address in private net.";
                }
                leaf ipv4-source-masklength {
                  type uint8 {
                    range "0..32";
```

```
                }
                description
                  "Source mask length for ipv4 addresses in
                   the group address pool in private net.";
              }
            }
            case acl {
              description "Use the type of acl";
              leaf group-acl-ipv4 {
                type string {
                  length "1..32";
                }
                description
                  "Specify the (s, g) entry on which the
                   S-PMSI tunnel takes effect.
                   The value is an integer ranging from 3000
                   to 3999 or a string of 32 case-sensitive
                   characters. If no value is specified, the
                   switch-group address pool takes effect on
                   all (s, g).";
              }
            }
          }
        }
      }

      grouping mvpn-spmsi-tunnel-config-ipv6 {
        description "Data mdt for rosen mvpn and S-PMSI for ng mvpn";

        container mvpn-spmsi-tunnels {
          description "S-PMSI tunnel configuration";
          leaf switch-delay-time {
            type uint8 {
              range "3..60";
            }
            units seconds;
            default "5";
            description
            "Delay for switching from the I-PMSI to
             the S-PMSI. The value is an integer
             ranging from 3 to 60, in seconds. ";
          }
          leaf switch-back-holddown-time {
            type uint16 {
              range "0..512";
            }
            units seconds;
            default "60";
```

```
            description
              "Delay for switching back from the S-PMSI
               to the I-PMSI. The value is an integer
               ranging from 0 to 512, in seconds. ";
          }
          leaf tunnel-limit {
            type uint16 {
              range "1..1024";
            }
            description
              "Maximum number of s-pmsi tunnels allowed.";
          }

          list mvpn-spmsi-tunnel {
            key "tunnel-type";
            description "S-PMSI tunnel parameter configuration";

            uses mvpn-spmsi-tunnel-basic-config;

            choice address-mask-or-acl {
              description "Type of define private net multicast address range";
              case address-mask {
                description "Use the type of address and mask";
                leaf ipv6-group-addr {
                  type inet:ipv6-address;
                  description
                    "Start and end ipv6 addresses of the group
                     address in private net.";
                }
                leaf ipv6-groupmasklength {
                  type uint8 {
                    range "8..128";
                  }
                  description
                    "Group mask length for ipv6 addresses in
                     the group address pool in private net.";
                }
                leaf ipv6-source-addr {
                  type inet:ipv6-address;
                  description
                    "Start and end ipv6 addresses of the source
                     address in private net.";
                }
                leaf ipv6-source-masklength {
                  type uint8 {
                    range "0..128";
                  }
                  description
                    "Source mask length for ipv6 addresses in
```

```
                    the group address pool in private net.";
                }
              }
              case acl {
                description "Use the type of acl";
                leaf group-acl-ipv6 {
                  type string {
                    length "1..32";
                  }
                  description
                    "Specify the (s, g) entry on which the
                     S-PMSI tunnel takes effect.
                     The value is an integer ranging from 3000
                     to 3999 or a string of 32 case-sensitive
                     characters. If no value is specified, the
                     switch-group address pool takes effect on
                     all (s, g).";
                }
              }
            }
          }
        }
      }

      grouping mvpn-pmsi-state {
        description "PMSI tunnel operational state information";
        leaf tunnel-type {
          type enumeration {
            enum invalid {
              value "0";
              description "invalid";
            }
            enum p2mp-te {
              value "1";
              description "p2mp-te";
            }
            enum p2mp-mldp {
              value "2";
              description "p2mp-mldp";
            }
            enum pim-ssm {
              value "3";
              description "pim-ssm";
            }
            enum pim-sm {
              value "4";
              description "pim-sm";
            }
            enum bidir-pim {
```

```
            value "5";
            description "bidir-pim";
          }
          enum ingress-replication {
            value "6";
            description "ingress-replication";
          }
          enum mp2mp-mldp {
            value "7";
            description "mp2mp-mldp";
          }
        }
        description "PMSI tunnel type.";
      }
      choice pmsi-tunnel-attribute {
        description "PMSI tunnel operational state information for each type";
        case p2mp-te {
          description "P2mp te tunnel";
          leaf te-p2mp-id {
            type uint16 {
              range "0..65535";
            }
            default "0";
            description "P2mp id of the p2mp tunnel.";
          }
          leaf te-tunnel-id {
            type uint16 {
              range "1..65535";
            }
            description "Id of the p2mp tunnel.";
          }
          leaf te-extend-tunnel-id {
            type uint16 {
              range "1..65535";
            }
            description "P2mp extended tunnel interface id.";
          }
        }
        case p2mp-mldp {
          description "P2mp mldp tunnel";
          leaf mldp-root-addr {
            type inet:ip-address;
            description "Ip address of the root of a p2mp ldp lsp.";
          }
          leaf mldp-lsp-id {
            type string {
              length "1..256";
            }
            description "P2mp ldp lsp id.";
```

```
            }
          }
          case pim-ssm {
            description "Pim ssm tunnel";
            leaf ssm-group-addr {
              type inet:ip-address;
              description "Group address for pim ssm";
            }
          }
          case pim-sm {
            description "Pim sm tunnel";
            leaf sm-group-addr {
              type inet:ip-address;
              description "Group address for pim sm";
            }
          }
          case bidir-pim {
            description "Bidir pim tunnel";
            leaf bidir-group-addr {
              type inet:ip-address;
              description "Group address for bidir-pim";
            }
          }
          case ingress-replication {
            description "Ingress replication p2p tunnel";
          }
          case mp2mp-mldp {
            description "mp2mp mldp tunnel";
          }
        }
        leaf tunnel-role {
          type enumeration {
            enum none {
              value "0";
              description "none";
            }
            enum root {
              value "1";
              description "root";
            }
            enum leaf {
              value "2";
              description "leaf";
            }
            enum root-and-leaf {
              value "3";
              description "root-and-leaf";
            }
          }
```

```
          description "Role of a tunnel node.";
        }
      }

      grouping mvpn-pmsi-entry-ipv4 {
        description
          "Multicast entries in ipv4 mvpn referenced the pmsi tunnel";
        container mvpn-pmsi-sg-ref-ipv4s {
          description
            "Multicast entries in ipv4 mvpn referenced the pmsi tunnel";
          list mvpn-pmsi-sg-ref-ipv4 {
            key "ipv4-source-address ipv4-group-address";
            description "Ipv4 source and group address";
            leaf ipv4-source-address {
              type inet:ipv4-address;
              description "Source address in I-PMSI or S-PMSI for ipv4.";
            }
            leaf ipv4-group-address {
              type inet:ipv4-address;
              description "Group address in I-PMSI or S-PMSI for ipv4.";
            }
          }
        }
      }


      grouping mvpn-pmsi-entry-ipv6 {
        description
          "Multicast entries in ipv6 mvpn referenced the pmsi tunnel";
        container mvpn-pmsi-sg-ref-ipv6s {
          description
            "Multicast entries in ipv6 mvpn referenced the pmsi tunnel";
          list mvpn-pmsi-sg-ref-ipv6 {
            key "ipv6-source-address ipv6-group-address";
            description "Ipv6 source and group address";
            leaf ipv6-source-address {
              type inet:ipv6-address;
              description "Source address in I-PMSI or S-PMSI for ipv6.";
            }
            leaf ipv6-group-address {
              type inet:ipv6-address;
              description "Group address in I-PMSI or S-PMSI for ipv6.";
            }
          }
        }
      }

      grouping mvpn-ipmsi-tunnel-state-ipv4 {
        description
```

```
        "Default mdt or I-PMSI operational state information";
      container mvpn-ipmsi-tunnel-info {
        description
          "Default mdt or I-PMSI operational state information";
        uses mvpn-pmsi-state;
        uses mvpn-pmsi-entry-ipv4;
      }
    }

    grouping mvpn-ipmsi-tunnel-state-ipv6 {
      description
        "Default mdt or I-PMSI operational state information";
      container mvpn-ipmsi-tunnel-info {
        description
          "Default mdt or I-PMSI operational state information";
        uses mvpn-pmsi-state;
        uses mvpn-pmsi-entry-ipv6;
      }
    }

    grouping mvpn-spmsi-tunnel-state-ipv4 {
      description
        "Data mdt or S-PMSI operational state information";
      container mvpn-spmsi-tunnel-ipv4-infos {
        description
          "Data mdt or S-PMSI operational state information";
        list mvpn-spmsi-tunnel-ipv4-info {
          key "tunnel-type";
          description
            "Data mdt or S-PMSI operational state information";
          uses mvpn-pmsi-state;
          uses mvpn-pmsi-entry-ipv4;
        }
      }
    }

    grouping mvpn-spmsi-tunnel-state-ipv6 {
      description
        "Data mdt or S-PMSI operational state information";
      container mvpn-spmsi-tunnel-ipv6-infos {
        description
          "Data mdt or S-PMSI operational state information";
        list mvpn-spmsi-tunnel-ipv6-info {
          key "tunnel-type";
          description
            "Data mdt or S-PMSI operational state information";
          uses mvpn-pmsi-state;
          uses mvpn-pmsi-entry-ipv6;
        }
```

```
      }
    }

    grouping l3vpn-mvrf-params {
      description
        "Specify multicast vrf parameters and provide
         multicast vrf operational state information";
      container ipv4 {
        description
          "Specify multicast ipv4 vrf parameters and provide
           multicast ipv4 vrf operational state information";
          container mvpn {
            description "Specify multicast ipv4 vrf parameters";
            uses mvpn-instance-config;
            uses mvpn-vpn-targets;
            uses mvpn-ipmsi-tunnel-config;
            uses mvpn-spmsi-tunnel-config-ipv4;
          }
          container mvpn-state {
            config "false";
            description
              "Multicast ipv4 vrf operational state information";
            uses mvpn-instance-config;
            uses mvpn-vpn-targets;
            uses mvpn-ipmsi-tunnel-config;
            uses mvpn-spmsi-tunnel-config-ipv4;
            uses mvpn-ipmsi-tunnel-state-ipv4;
            uses mvpn-spmsi-tunnel-state-ipv4;
          }
        }
      container ipv6 {
        description
          "Ipv6 address family specific multicast vrf parameters and
           multicast vrf operational state information";
          container mvpn {
            description "Ipv6 address family specific multicast vrf parameters
."";
            uses mvpn-instance-config;
            uses mvpn-vpn-targets;
            uses mvpn-ipmsi-tunnel-config;
            uses mvpn-spmsi-tunnel-config-ipv6;
          }
          container mvpn-state {
            config "false";
            description
              "Ipv6 address family multicast vrf operational state information
."";
            uses mvpn-instance-config;
            uses mvpn-vpn-targets;
            uses mvpn-ipmsi-tunnel-config;
            uses mvpn-spmsi-tunnel-config-ipv6;
```

```
              uses mvpn-ipmsi-tunnel-state-ipv6;
              uses mvpn-spmsi-tunnel-state-ipv6;
            }
          }
        }


        augment "/ni:network-instances/ni:network-instance" {
          description
            "Augment network instance container for per multicast VRF config";
          container l3vpn {
            //when "../type='rt:vrf-routing-instance'" {
            //description
            //  "This container is only valid for multicast vrf
            //   routing instance.";
            //}
            description
              "Configuration of multicast vpn specific parameters and
               operational state of multicast vpn specific parameters";
            uses l3vpn-mvrf-params;
          }
        }
    }
    <CODE ENDS>
```

5. Security Considerations

The data model defined does not introduce any security implications. This draft does not change any underlying security issues inherent in [RFC8022].

6. IANA Considerations

TBD

7. References

7.1. Normative References

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-05 (work in progress), October 2015.

[RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, November 2016.

7.2. Informative References

[RFC6037] Rosen, E., Cai, Y., and IJ. Wijnands, "Cisco Systems' Solution for Multicast in BGP/MPLS IP VPNs", RFC 6037, October 2010.

[RFC6513] Rosen, E. and R. Aggarwal, "Multicast in MPLS/BGP IP VPNs", RFC 6513, February 2012.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, February 2012.

[RFC7246] IJ. Wijnands, P. Hitchen, N. Leymann, W. Henderickx, A. Gulko and J. Tantsura, " Multipoint Label Distribution Protocol In-Band Signaling in a Virtual Routing and Forwarding (VRF) Table Context ", RFC 7246, June 2014.

[RFC7900] Y. Rekhter, E. Rosen, R. Aggarwal, Arktan, Y. Cai and T. Morin, " Extranet Multicast in BGP/IP MPLS VPNs ", RFC 7900, June 2016.

8. Acknowledgments

The authors would like to thank Anish Peter, Stig Venaas for their valuable contributions.

Authors' Addresses

Yisong Liu
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: liuyisong@huawei.com

Feng Guo
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: guofeng@huawei.com

Xufeng Liu
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

Email: Xufeng_Liu@jabil.com

Robert Kebler
Juniper Networks
10 Technology Park Drive
Westford, MA 01886
USA

Email: rkebler@juniper.net

Mahesh Sivakumar
Cisco Systems, Inc
510 McCarthy Blvd
Milpitas, California 95035
USA

Email: masivaku@cisco.com

BESS Workgroup
Internet Draft
Intended status: Standards Track

J. Rabadan, Ed.
S. Sathappan
Nokia

S. Boutros
VMware

T. Przygienda
W. Lin
J. Drake
Juniper Networks

A. Sajassi
S. Mohanty
Cisco Systems

Expires: June 22, 2017

December 19, 2016

Preference-based EVPN DF Election
draft-rabadan-bess-evpn-pref-df-02

Abstract

RFC7432 defines the Designated Forwarder (DF) in (PBB-)EVPN networks as the PE responsible for sending broadcast, multicast and unknown unicast traffic (BUM) to a multi-homed device/network in the case of an all-active multi-homing ES, or BUM and unicast in the case of single-active multi-homing.

The DF is selected out of a candidate list of PEs that advertise the Ethernet Segment Identifier (ESI) to the EVPN network, according to the 'service-carving' algorithm.

While 'service-carving' provides an efficient and automated way of selecting the DF across different EVIs or ISIDs in the ES, there are some use-cases where a more 'deterministic' and user-controlled method is required. At the same time, Service Providers require an easy way to force an on-demand DF switchover in order to carry out some maintenance tasks on the existing DF or control whether a new active PE can preempt the existing DF PE.

This document proposes an extension to the current RFC7432 DF election procedures so that the above requirements can be met.

Status of this Memo

This Internet-Draft is submitted in full conformance with the

Copyright Notice

Table of Contents

1. Problem Statement 3
2. Solution requirements 3
3. EVPN BGP Attributes for Deterministic DF Election 4
4. Solution description 5
4.1 Use of the Preference algorithm 5
4.2 Use of the Preference algorithm in RFC7432 Ethernet-Segments 7
4.3 The Non-Revertive option 7
5. Conclusions 10

11. Conventions used in this document 10
12. Security Considerations 10
13. IANA Considerations 11
15. References 11
15.1 Normative References 11
15.2 Informative References 11
16. Acknowledgments 11
17. Contributors 11
17. Authors' Addresses 11

1. Problem Statement

RFC7432 defines the Designated Forwarder (DF) in (PBB-)EVPN networks as the PE responsible for sending broadcast, multicast and unknown unicast traffic (BUM) to a multi-homed device/network in the case of an all-active multi-homing ES or BUM and unicast traffic to a multi-homed device or network in case of single-active multi-homing.

The DF is selected out of a candidate list of PEs that advertise the Ethernet Segment Identifier (ESI) to the EVPN network and according to the 'service-carving' algorithm.

While 'service-carving' provides an efficient and automated way of selecting the DF across different EVIs or ISIDs in the ES, there are some use-cases where a more 'deterministic' and user-controlled method is required. At the same time, Service Providers require an easy way to force an on-demand DF switchover in order to carry out some maintenance tasks on the existing DF or control whether a new active PE can preempt the existing DF PE.

This document proposes an extension to the current RFC7432 DF election procedures so that the above requirements can be met.

2. Solution requirements

This document proposes an extension of the RFC7432 'service-carving' DF election algorithm motivated by the following requirements:

a) The solution MUST provide an administrative preference option so that the user can control in what order the candidate PEs may become DF, assuming they are all operationally ready to take over.

b) This extension MUST work for RFC7432 Ethernet Segments (ES) and virtual ES, as defined in [vES].

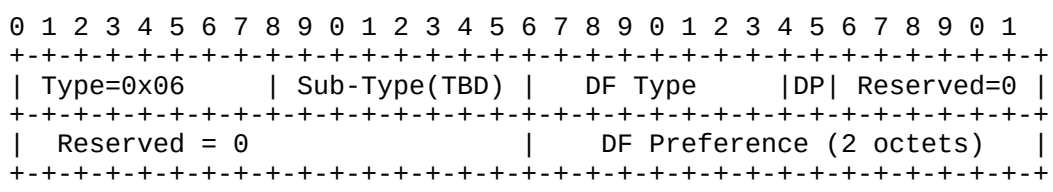
c) The user MUST be able to force a PE to preempt the existing DF for a given EVI/ISID without re-configuring all the PEs in the ES.

d) The solution SHOULD allow an option to NOT preempt the current DF, even if the former DF PE comes back up after a failure. This is also known as "non-revertive" behavior, as opposed to the RFC7432 DF election procedures that are always revertive.

e) The solution MUST work for single-active and all-active multi-homing Ethernet Segments.

3. EVPN BGP Attributes for Deterministic DF Election

This solution reuses and extends the DF Election Extended Community defined in [EVPN-HRW-DF] that is advertised along with the ES route:



Where the following fields are re-defined as follows:

o DF Type can have the following values:

- Type 0 - Default, mod based DF election as per RFC7432.
- Type 1 - HRW algorithm as per [EVPN-HRW-DF]
- Type 2 - Preference algorithm (this document)

o DP or ’Don’t Preempt’ bit, determines if the PE advertising the ES route requests the remote PEs in the ES not to preempt it as DF. The default value is DP=0, which is compatible with the current ’preempt’ or ’revertive’ behavior in RFC7432. The DP bit SHOULD be ignored if the DF Type is different than 2.

o DF Preference defines a 2-octet value that indicates the PE preference to become the DF in the ES. The default value MUST be 32767. This value is the midpoint in the allowed Preference range of values, which gives the operator the flexibility of choosing a significant number of values, above or below the default Preference.

4. Solution description

Figure 1 illustrates an example that will be used in the description of the solution.

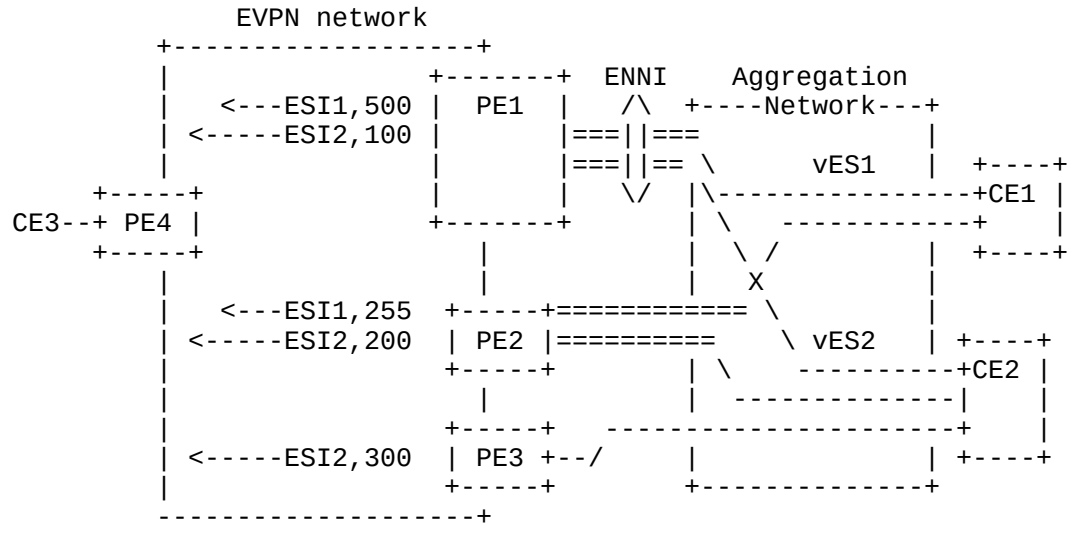


Figure 1 ES and Deterministic DF Election

Figure 1 shows three PEs that are connecting EVCs coming from the Aggregation Network to their EVIs in the EVPN network. CE1 is connected to vES1 - that spans PE1 and PE2 - and CE2 is connected to vES2, that is defined in PE1, PE2 and PE3.

If the algorithm chosen for vES1 and vES2 is type 2, i.e. Preference-based, the PEs may become DF irrespective of their IP address and based on an administrative Preference value. The following sections provide some examples of the new defined procedures and how they are applied in the use-case in Figure 1.

4.1 Use of the Preference algorithm

Assuming the operator wants to control - in a flexible way - what PE becomes the DF for a given vES and the order in which the PEs become DF in case of multiple failures, the following procedure may be used:

a) vES1 and vES2 are now configurable with three optional parameters that are signaled in the DF Election extended community. These parameters are the Preference, Preemption option (or "Don’t

Preempt Me" option) and DF algorithm type. We will represent these parameters as [Pref,DP,type]. Let’s assume vES1 is configured as [500,0,Pref] in PE1, and [255,0,Pref] in PE2. vES2 is configured as [100,0,Pref], [200,0,Pref] and [300,0,Pref] in PE1, PE2 and PE3 respectively.

b) The PEs will advertise an ES route for each vES, including the 3 parameters in the DF Election Extended Community.

c) According to RFC7432, each PE will wait for the DF timer to expire before running the DF election algorithm. After the timer expires, each PE runs the Preference-based DF election algorithm as follows:

   o The PE will check the DF type in each ES route, and assuming all the ES routes are consistent in this DF type and the value is 2 (Preference-based), the PE will run the new extended procedure. Otherwise, the procedure will fall back to RFC7432 ’service-carving’.

   o In this extended procedure, each PE builds a list of candidate PEs, ordered based on the Preference. E.g. PE1 will build a list of candidate PEs for vES1 ordered by the Preference, from high to low: PE1>PE2. Hence PE1 will become the DF for vES1. In the same way, PE3 becomes the DF for vES2.

d) Note that, by default, the Highest-Preference is chosen for each ES or vES, however the ES configuration can be changed to the Lowest-Preference algorithm as long as this option is consistent in all the PEs in the ES. E.g. vES1 could have been explicitly configured as type Preference-based with Lowest-Preference, in which case, PE2 would have been the DF.

e) Assuming some maintenance tasks had to be executed on PE3, the operator could set vES2’s preference to e.g. 50 so that PE2 is forced to take over as DF for vES2. Once the maintenance on PE3 is over, the operator could decide to leave the existing preference or configure the old preference back.

f) In case of equal Preference in two or more PEs in the ES, the tie-breakers will be the DP bit and the lowest IP PE in that order. For instance:

   o If vES1 parameters were [500,0,Pref] in PE1 and [500,1,Pref] in PE2, PE2 would be elected due to the DP bit.

   o If vES1 parameters were [500,0,Pref] in PE1 and [500,0,Pref] in PE2, PE1 would be elected, assuming PE1’s IP address is lower

than PE2's.

g) The Preference is an administrative option that MUST be configured on a per-ES basis from the management plane, but MAY also be dynamically changed based on the use of local policies. For instance, on PE1, ES1's Preference can be lowered from 500 to 100 in case the bandwidth on the ENNI port is decreased a 50% (that could happen if e.g. the 2-port LAG between PE1 and the Aggregation Network loses one port). Policies MAY also trigger dynamic Preference changes based on the PE's bandwidth availability in the core, of specific ports going operationally down, etc. The definition of the actual local policies is out of scope of this document. The default Preference value is 32767.

4.2 Use of the Preference algorithm in RFC7432 Ethernet-Segments

While the Preference-based DF type described in section 4.1 is typically used in virtual ES scenarios where there is normally an individual EVI per vES, the existing RFC7432 definition of ES allows potentially up to thousands of EVIs on the same ES. If this is the case, and the operator still wants to control who the DF is for a given EVI, the use of the Preference-based DF type can also provide the desired level of load balancing.

In this type of scenarios, the ES is configured with an administrative Preference value, but then a range of EVI/ISIDs can be defined to use the Highest-Preference or the Lowest-Preference depending on the desired behavior. With this option, the PE will build a list of candidate PEs ordered by the Preference, however the DF for a given EVI/ISID will be determined by the local configuration.

For instance:

- Assuming ES3 is defined in PE1 and PE2, PE1 may be configured as [500,0,Preference] for ES3 and PE2 as [100,0,Preference].
- In addition, assuming vlan-based service interfaces, the PEs will be configured with (vlan/ISID-range,high_or_low), e.g. (1-2000,high) and (2001-4000, low).
- This will result in PE1 being DF for EVI/ISIDs 1-2000 and PE2 being DF for EVI/ISIDs 2001-4000.

4.3 The Non-Revertive option

As discussed in section 2(d), an option to NOT preempt the existing

DF for a given EVI/ISID is required and therefore added to the DF Election extended community. This option will allow a non-revertive behavior in the DF election.

Note that, when a given PE in an ES is taken down for maintenance operations, before bringing it back, the Preference may be changed in order to provide a non-revertive behavior. The DP bit and the mechanism explained in this section will be used for those cases when a former DF comes back up without any controlled maintenance operation, and the non-revertive option is desired in order to avoid service impact.

In Figure 1, we assume that based on the Highest-Pref, PE3 is the DF for ESI2.

If PE3 has a link, EVC or node failure, PE2 would take over as DF. If/when PE3 comes back up again, PE3 will take over, causing some unnecessary packet loss in the ES.

The following procedure avoids preemption upon failure recovery (please refer to Figure 1):

1) A new "Don’t Preempt Me" parameter is defined on a per-PE per-ES basis, as described in section 3. If "Don’t Preempt Me" is disabled (default behavior) the advertised DP bit will be 0. If "Don’t Preempt Me" is enabled, the ES route will be advertised with DP=1 ("Don’t Preempt Me").

2) Assuming we want to avoid ’preemption’, the three PEs are configured with the "Don’t Preempt Me" option. Note that each PE individually MAY be configured with different preemption value. In this example, we assume ESI2 is configured as ’DP=enabled’ in the three PEs.

3) Assuming EVI1 uses Highest-Pref in vES2 and EVI2 uses Lowest-Pref, when vES2 is enabled in the three PEs, the PEs will exchange the ES routes and select PE3 as DF for EVI1 (due to the Highest-Pref type), and PE1 as DF for EVI2 (due to the Lowest-Pref).

4) If PE3’s vES2 goes down (due to EVC failure - detected by OAM, or port failure or node failure), PE2 will become the DF for EVI1. No changes will occur for EVI2.

5) When PE3’s vES2 comes back up, PE3 will start a boot-timer (if booting up) or hold-timer (if the port or EVC recovers). That timer will allow some time for PE3 to receive the ES routes from PE1 and PE2. PE3 will then:

o Select two "reference-PEs" among the ES routes in the vES, the "Highest-PE" and the "Lowest-PE":

  - The Highest-PE is the PE with higher Preference, using the DP bit first (with DP=1 being better) and, after that, the lower PE-IP address as tie-breakers. PE3 will select PE2 as Highest-PE over PE1, since, when comparing [Pref,DP,PE-IP], [200,1,PE2-IP] wins over [100,1,PE1-IP].

  - The Lowest-PE is the PE with lower Preference, using the DP bit first (with DP=1 being better) and, after that, the lower PE-IP address as tie-breakers. PE3 will select PE1 as Lowest-PE over PE2, since [100,1,PE1-IP] wins over [200,1,PE2-IP].

  - Note that if there were only one remote PE in the ES, Lowest and Highest PE would be the same PE.

o Check its own administrative Pref and compares it with the one of the Highest-PE and Lowest-PE that have DP=1 in their ES routes. Depending on this comparison PE3 will send the ES route with a [Pref,DP] that may be different from its administrative [Pref,DP]:

  - If PE3’s Pref value is higher than the Highest-PE’s, PE3 will send the ES route with an ’in-use’ operational Pref equal to the Highest-PE’s and DP=0.

  - If PE3’s Pref value is lower than the Lowest-PE’s, PE3 will send the ES route with an ’in-use’ operational Preference equal to the Lowest-PE’s and DP=0.

  - If PE3’s Pref value is neither higher nor lower than the Highest-PE’s or the Lowest-PE’s respectively, PE3 will send the ES route with its administrative [Pref,DP]=[300,1].

  - In this example, PE3’s administrative Pref=300 is higher than the Highest-PE with DP=1, that is, PE2 (Pref=200). Hence PE3 will inherit PE2’s preference and send the ES route with an operational ’in-use’ [Pref,DP]=[200,0].

Note that, a PE will always send DP=0 as long as the advertised Pref is the ’in-use’ operational Pref (as opposed to the ’administrative’ Pref).

This ES route update sent by PE3 (with [200,0,PE3-IP]) will not cause any DF switchover for any EVI/ISID. PE2 will continue being DF for EVI1. This is because the DP bit will be used as a tie-

breaker in the DF election. That is, if a PE has two candidate PEs with the same Pref, it will pick up the one with DP=1. There are no DF changes for EVI2 either.

6) Subsequently, if PE2 fails, upon receiving PE2's ES route withdrawal, PE3 and PE1 will go through the process described in (5) to select new Highest and Lowest-PEs (considering their own active ES route) and then they will run the DF Election.

   - If a PE selects itself as new Highest or Lowest-PE and it was not before, the PE will then compare its operational 'in-use' Pref with its administrative Pref. If different, the PE will send an ES route update with its administrative Pref and DP values. In the example, PE3 will be the new Highest-PE, therefore it will send an ES route update with [Pref,DP]=[300,1].

   - After running the DF Election, PE3 will become the new DF for EVI1. No changes will occur for EVI2.

5. Conclusions

Service Providers are seeking for options where the DF election can be controlled by the user in a deterministic way and with a non-revertive behavior. This document defines the use of a Preference algorithm that can be configured and used in a flexible manner to achieve those objectives.

11. Conventions used in this document

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 RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

12. Security Considerations

This section will be added in future versions.

13. IANA Considerations

This document solicits the allocation of DF type = 2 in the registry created by [EVPN-HRW-DF] for the DF type field.

15. References

15.1 Normative References

[RFC7432]Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <http://www.rfc-editor.org/info/rfc7432>.

15.2 Informative References

[vES] Sajassi et al. "EVPN Virtual Ethernet Segment", draft-sajassi-bess-evpn-virtual-eth-segment-01, work-in-progress, July 6, 2015.

[EVPN-HRW-DF] Mohanty S. et al. "A new Designated Forwarder Election for the EVPN", draft-mohanty-bess-evpn-df-election-02, work-in-progress, October 19, 2015.

16. Acknowledgments

17. Contributors

In addition to the authors listed, the following individuals also contributed to this document:

Kiran Nagaraj, Nokia
Vinod Prabhu, Nokia
Selvakumar Sivaraj, Juniper

17. Authors’ Addresses

Jorge Rabadan
Nokia
777 E. Middlefield Road
Mountain View, CA 94043 USA
Email: jorge.rabadan@nokia.com

Senthil Sathappan
Alcatel-Lucent
Email: senthil.sathappan@nokia.com

Tony Przygienda
Juniper Networks, Inc.
Email: prz@juniper.net

John Drake
Juniper Networks, Inc.
Email: jdrake@juniper.net

Wen Lin
Juniper Networks, Inc.
Email: wlin@juniper.net

Ali Sajassi
Cisco Systems, Inc.
Email: sajassi@cisco.com

Satya Ranjan Mohanty
Cisco Systems, Inc.
Email: satyamoh@cisco.com

Sami Boutros
VMware, Inc.
Email: sboutros@vmware.com

BESS Workgroup
Internet Draft
Intended status: Standards Track

J. Rabadan, Ed.
A. Simpson, Ed.
Nokia

J. Uttaro
AT&T

Expires: September 14, 2017

March 13, 2017

EVPN Path Attribute Propagation
draft-rs-bess-evpn-attr-prop-00

Abstract

EVPN is being actively used to provide tenant inter-subnet-forwarding in DC networks, as described in [IP-PREFIX] and [INTER-SUBNET]. When those tenant networks are interconnected to vpn-ipv4/vpn-ipv6 or ipv4/ipv6 BGP networks, there is a need for the EVPN BGP Path Attributes to be seamlessly propagated so that the receiver PE or NVE can consider the original EVPN Attributes in its path calculations. This document analyses the use-cases, requirements and rules based on which the BGP Path Attributes should be propagated between EVPN and other BGP families.

This Internet-Draft will expire on September 14, 2017.

Table of Contents

1. Problem Statement 2
2. EVPN Path Attribute Propagation Use Cases 3
2.1 DCI using a Different Administrative Domain 3
2.2 DCI within the Same Administrative Domain 4
2.3 DCI using a Public IP Network 5
3. Solution Requirements 6
4. Solution Description 6
4.1 EVPN Path Attribute No-Propagation-Mode 6
4.2 EVPN Path Attribute Propagation Tunnel-Mode 7
4.3 EVPN Path Attribute Propagation Uniform-Mode 8
4.4 Path Selection across EVPN and IP-VPN 9
5. Deployment Examples 9
6. Conclusions 9
6. Conventions used in this document 10
7. Security Considerations 10
8. IANA Considerations 10
9. Terminology 10
9. References 11
9.1 Normative References 11
9.2 Informative References 11
10. Acknowledgments 11
11. Contributors 11
17. Authors' Addresses 11

1. Problem Statement

EVPN is being actively used to provide tenant inter-subnet-forwarding in DC networks, as described in [IP-PREFIX] and [INTER-SUBNET]. When those tenant networks are interconnected to vpn-ipv4/vpn-ipv6 or ipv4/ipv6 BGP networks, there is a need for the EVPN BGP Path Attributes to be seamlessly propagated so that the receiver PE or NVE can consider the original EVPN Attributes in its path calculations. This document analyses the use-cases, requirements and rules based on which the BGP Path Attribute propagation should be propagated between EVPN and other BGP families.

EVPN supports the advertisement of ipv4 or ipv6 prefixes in two different route types:

- Route Type 2 - MAC/IP route (only for /32 and /128 host routes), as described by [INTER-SUBNET].

- Route Type 5 - IP Prefix route, as described by [IP-PREFIX].

This proposal describes how the BGP Path Attributes sent along those routes should be propagated to other BGP families being used to advertise tenant IP-Prefixes, such as VPN-IPv4 (AFI/SAFI 1/128), VPN-IPv6 (AFI/SAFI 2/128), IPv4 (AFI/SAFI 1/1) or IPv6 (AFI/SAFI 2/1).

2. EVPN Path Attribute Propagation Use Cases

The following Data Center Interconnect (DCI) use-cases have been identified and will be used as a reference in this document.

2.1 DCI using a Different Administrative Domain

The assumption in this use-case is that Data Centers (DCs) are connected to other DCs by provider networks that are managed by different administrative entities. While EVPN is used within the DCs to exchange IP Prefixes, the provider interconnect network uses IP-VPN to exchange IP reachability. DC Gateway pairs DGW1 and DGW2 provide a Boundary Router (BR) function between the EVPN and IP-VPN families.

As an example, let’s assume NVE1 and NVE2 both advertise an "anycast" prefix A/32. NVE1 uses a Route Type 2 (RT2) or MAC/IP route to encode the A/32 prefix, while NVE1 uses a Route Type 5 (RT5) or IP-Prefix route to encode A/32. DGW1 routers import the routes into the IP-VRF routing table and re-advertise them to the IP-VPN network using a different RD, probably different route-target and their own Next-Hop. DGW2 routers do the opposite translation and re-advertise the host routes using EVPN RT5s. NVE4 uses a PE-CE eBGP session to advertise the host routes to the CE.

While NVEs at DC1 and DC2 set the proper Path Attributes, for example LOCAL_PREFERENCE and Communities 'red' and 'blue', so that NVEs within the DCs can make the right path selection, those Path Attributes are lost when the routes are re-generated at the Boundary Routers (BRs). When the EVPN routes arrive at NVE3 or CE, the path selection cannot be influenced as intended by the NVEs that originated the routes. A set of procedures is needed so that the IP-VPN provider network tunnels all the relevant original EVPN Path Attributes transparently up to the destination EVPN DC.

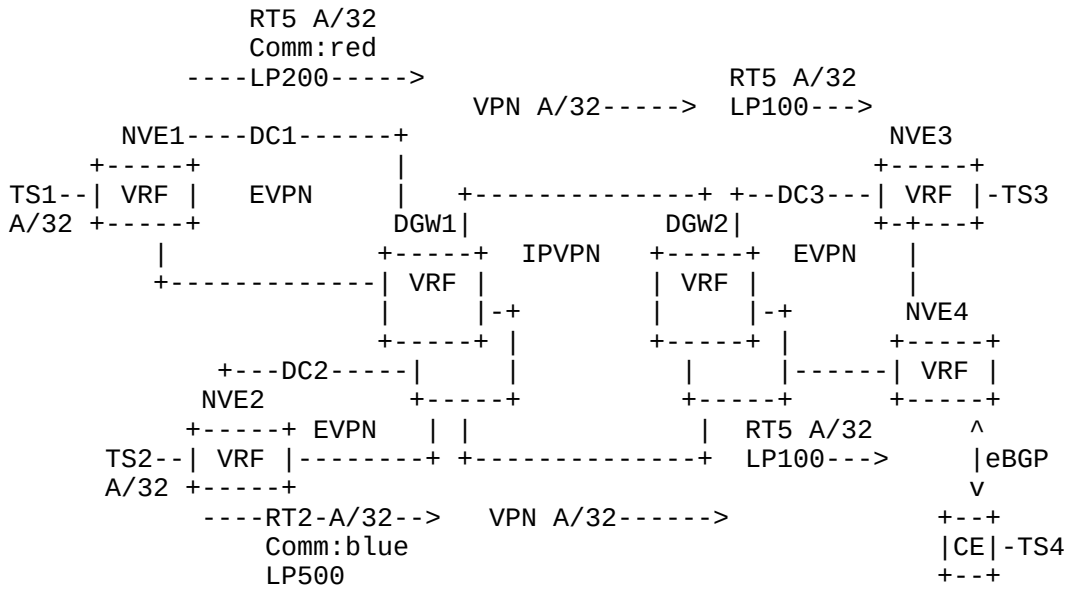


Figure 1 DCI using a Different Administrative Domain

2.2 DCI within the Same Administrative Domain

Use-case 2.1 assumed that EVPN DCs were connected using an IP-VPN provider network and there was a need to "tunnel" the original EVPN Path Attributes through the provider IP-VPN network up to the destination EVPN DC. In this section, the entire network is managed by the same entity. The destination PE2 in Figure 2 will receive the two host routes using VPN-IPv4 family directly, even though the routes were originated in the EVPN family.

Multiple models may exist for defining the over-arching VPN solution

defined by this family interaction:

a) In some cases, the BRs (Boundary Routers) need to re-originate the two host routes with the original EVPN Path Attributes (LOCAL_PREFERENCE and Communities in Figure 2) so that they are not lost for PE2's path calculations.

b) In some other cases, the EVPN domains are considered abstracted "CEs" for the IP-VPN network and the BRs just need to reinitialize the Path Attributes so that PE2 does not take the original EVPN Path Attribute into consideration for path calculations.

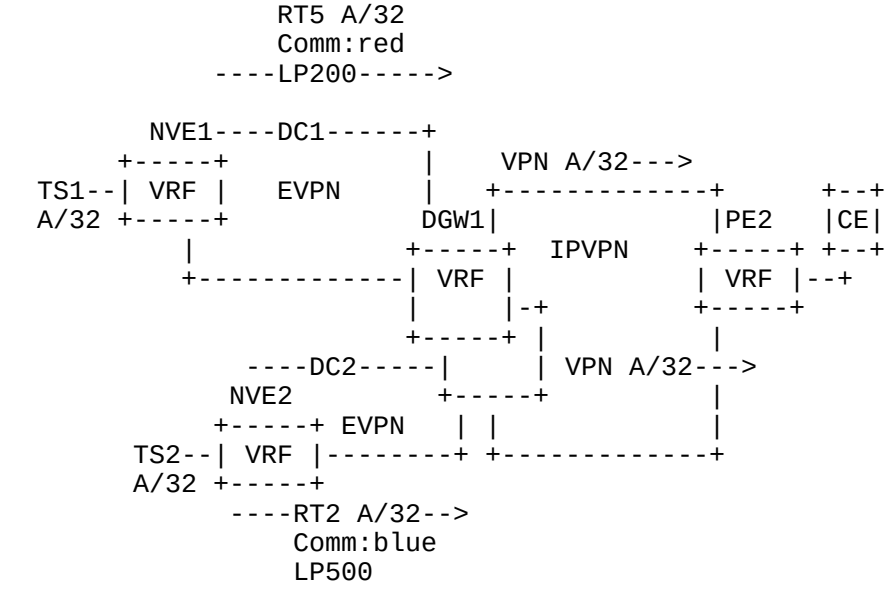


Figure 2 DCI within the Same Administrative Domain

The solution must support both models.

2.3 DCI using a Public IP Network

Figure 3 depicts a use-case similar to the one described in section 2.2, however the subnet RT5 is converted to an IPv4 route that gets propagated by BGP throughout a Public Network. As in the previous sections, when the route arrives at the CE router, the originating EVPN Path Attributes are lost. While this may be the desired situation in some cases, in some other cases there is a need to propagate the original EVPN Path Attributes all the way up to the CE router.

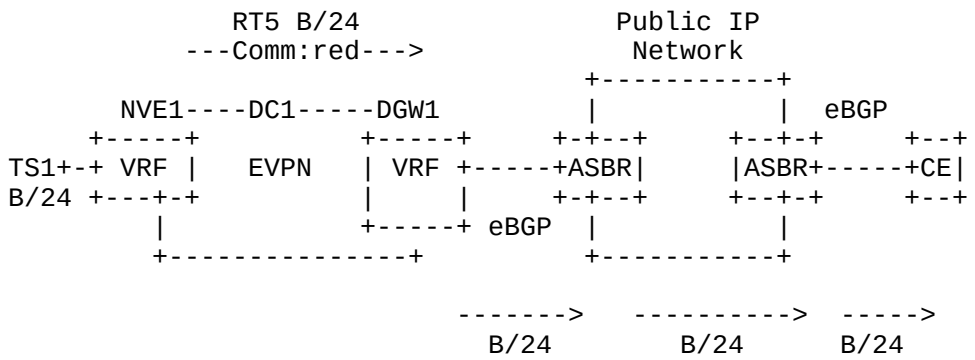


Figure 3 DCI using a Public IP Network

3. Solution Requirements

The following requirements have been identified for the Propagation of EVPN Path Attributes:

- o The EVPN Path Attribute Propagation solution MUST allow the propagation of path attributes among EVPN (SAFI 70), VPN (SAFI 128) and IP (SAFI 1) families, for IPv4 and IPv6 routes (AFIs 1 and 2).

- o The solution SHOULD allow the tunneling of the set of relevant Path Attributes between two BRs of the same family that are connected by another family. Figure 1 provides an example.

- o The solution SHOULD allow the propagation of certain key attributes (that are commonly used) between two different families. Figure 2 and 3 show two examples of cases where EVPN Path Attributes should keep accumulating or mapped rather than being tunneled.

4. Solution Description

This document proposes three Path Attribute Propagation Modes that satisfy the use-cases and requirements described in sections 2 and 3: No-Propagation-Mode, Tunnel-Mode and Uniform-Mode. In the following sections, the term "BR" or "Boundary Router" refers to the PE router that supports more than one SAFI to manage IP-prefixes in the same IP-VRF and is responsible for the Path Attribute Propagation across families.

4.1 EVPN Path Attribute No-Propagation-Mode

This is the default mode of operation. In this mode, the BR will

simply re-initialize the Path Attributes when re-advertising a route to a different SAFI, as though it would for direct or local IP-Prefixes. This model will meet the requirements in those use-cases where the EVPN domain is considered an "abstracted" CE and remote IP-VPN/IP PEs don't need to consider the original EVPN Attributes for path calculations.

4.2 EVPN Path Attribute Propagation Tunnel-Mode

In this mode, the Path Attributes are "tunneled" between an ingress and an egress BR. The ingress BR tunnels a set of path attributes for a given family across a provider network that uses a different family. It is typically used for DCs interconnected thru a different administrative domain, as in section 2.1.

The ATT_SET path attribute (defined in RFC6368) is used for this Path Attribute Propagation Tunnel-Mode as follows:

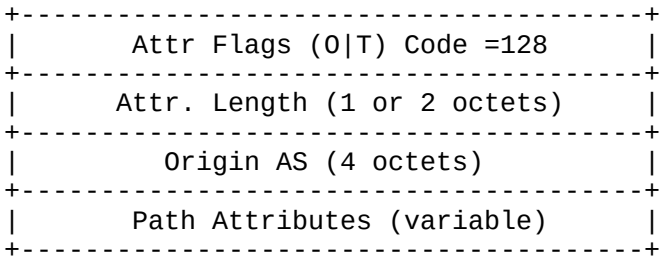


Figure 4 ATT_SET path attribute used for Tunnel-Mode

The following rules MUST be observed:

- These are the Path Attributes that MUST NOT be inserted in the ATT_SET by the ingress BR:
  - MP_REACH_NLRI
  - MP_UNREACH_NLRI
  - NEXT_HOP
  - PTA (PMSI Tunnel Attribute)
  - RFC5512 BGP Encapsulation extended community
  - Tunnel Encapsulation Attribute
  - EVPN-type (0x6) Extended Communities

- ATT_SET insertion rules at ingress BR:

  - IP Prefix routes (RT5 and RT2) learned by the ingress BR on the IP-VRF are imported and re-exported as a different AFI/SAFI with the ATT_SET added.

- The ATT_SET contains an exact copy of all received path attributes except for those that must not be propagated (see bullet above).

- The Origin AS in the attribute encodes the ASN of the exporting VRF.

- Once the ATTR_SET attribute is added to the route, the other path attributes are re-initialized to the basic values that would apply to an exported local/direct IP-VRF route (that is, a route without BGP attributes).

- Note that, compared to RFC6368, in this document ingress BR's IP-VRF does not need IBGP to the CE/NVE. EBGP is possible too. And also, the main focus of this document is EVPN to other families.

o ATT_SET extraction rules at the egress BR:

- The egress BR receiving the ATT_SET, imports the IP-Prefix routes into the IP-VRF, based on the IP-VRF import policies. Different RDs are expected for same routes received from different Next-Hops.

- The Path Attributes in ATT_SET replace the Path Attributes of the received route at the import process (so that the BGP decision process of each IP-VRF considers the original Path Attributes).

- The route, that is re-constructed from ATT_SET, is advertised to the BGP peers of the importing IP-VRF as per [RFC6368]:

  + If the peer is IBGP-based and ATT_SET's Origin AS matches the configured IP-VRF's AS, then the route is advertised "as-is" with Next-Hop-Self (and the original Path Attributes).

  + If the peer is IBGP-based and ATT_SET's Origin AS is different than the configured IP-VRF's AS, then the IBGP-specific Path Attributes are removed, and the ATT_SET Origin AS is prepended to the AS_PATH.

  + If the peer is EBGP-based, then the IBGP-specific Path Attributes are removed and the new AS_PATH will be composed of (ATT_SET Origin AS + received AS_PATH + configured IP-VRF's AS).

4.3 EVPN Path Attribute Propagation Uniform-Mode

In this mode, the BR simply keeps accumulating or mapping certain key

commonly used Path Attributes when re-advertising routes to a different family. This mode is typically used for DCs interconnected by the same administrative domain that manages the DCs, as in section 2.2.

The following rules MUST be observed by the BR when propagating Path Attributes:

- o The BR imports the routes in the IP-VRF and stores the original Path Attributes. Only the following set of Path Attributes SHOULD be propagated by the BR:
  - AS_PATH
  - IBGP-only Path Attributes: LOCAL_PREF, ORIGINATOR_ID, CLUSTER_ID
  - Communities, (non-EVPN) Extended Communities and Large Communities
- o When re-advertising a route to a destination family, the BR MUST copy the AS_PATH of the originating family and prepend the IP-VRF's AS (only for EBGP peers).
- o When re-advertising a route to IBGP peers, the BR MUST copy the IBGP-only Path Attributes from the originating family to the re-advertised route.
- o Communities, non-EVPN Extended Communities and Large Communities MUST be copied by the BR from the originating family.

Note: the need to include other Path Attributes, such as MED or AIGP, or modify the above behavior will be analyzed in future revisions of this document.

4.4 Path Selection across EVPN and IP-VPN

In some cases, an NVE/PE receives the same IP-Prefix from two different families, e.g. EVPN and IP-VPN. This section discusses how the NVE/PE should compare both routes and the rules of selection.

NOTE: this section will be completed in a future revision.

5. Deployment Examples

This section will be added in the next revision of the document.

6. Conclusions

This document describes the need to propagate EVPN Path Attributes so that NVE/PEs receiving IP-Prefix routes can select paths based on the Attributes that the advertising NVE/PE originally added to the route. In order to achieve that goal, three EVPN Path Attribute Propagation Modes are discussed:

a) No-Propagation-Mode
b) Tunnel-Mode
c) Uniform-Mode

While (a) is the default mode, (b) is required to preserve all the relevant EVPN Path Attributes in use-cases where different Administrative Domains provide connectivity; (c) provides a simple solution to propagate only certain commonly used Path Attributes that are typically used by providers.

This solution will help providers have a seamless EVPN integration in existing IP-VPN and IP networks.

6. Conventions used in this document

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 RFC-2119 [RFC2119].

In this document, these words will appear with that interpretation only when in ALL CAPS. Lower case uses of these words are not to be interpreted as carrying RFC-2119 significance.

In this document, the characters ">>" preceding an indented line(s) indicates a compliance requirement statement using the key words listed above. This convention aids reviewers in quickly identifying or finding the explicit compliance requirements of this RFC.

7. Security Considerations

This section will be added in future versions.

8. IANA Considerations

9. Terminology

BR: Boundary Router - refers to the router responsible for the Path Attribute Propagation.
RT2: Route Type 2 or MAC/IP route, as per [RFC7432].
RT5: Route Type 5 or IP-Prefix, as per [IP-PREFIX].

9. References

9.1 Normative References

[RFC7432]Sajassi, A., Ed., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, DOI 10.17487/RFC7432, February 2015, <http://www.rfc-editor.org/info/rfc7432>.

[RFC6368]Marques, P., Raszuk, R., Patel, K., Kumaki, K., and T. Yamagata, "Internal BGP as the Provider/Customer Edge Protocol for BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 6368, DOI 10.17487/RFC6368, September 2011, <http://www.rfc-editor.org/info/rfc6368>.

9.2 Informative References

[IP-PREFIX] Rabadan et al., "IP Prefix Advertisement in EVPN", draft-ietf-bess-evpn-prefix-advertisement-04, February, 2017.

[INTER-SUBNET] Sajassi et al., "IP Inter-Subnet Forwarding in EVPN", draft-ietf-bess-evpn-inter-subnet-forwarding-03.txt, work in progress, February, 2017

[ENCAP-ATT] Rosen et al., "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-03.txt, work in progress, November, 2016.

10. Acknowledgments

11. Contributors

17. Authors' Addresses

Jorge Rabadan
Nokia
777 E. Middlefield Road
Mountain View, CA 94043 USA
Email: jorge.rabadan@nokia.com

Adam Simpson
Nokia
Email: adam.1.simpson@nokia.com

Jim Uttaro
AT&T
Email: ju1738@att.com

BESS Working Group
Internet-Draft
Intended Status: Standards Track

Ali Sajassi
Gaurav Badoni
Dhananjaya Rao
Patrice Brissette
Cisco
John Drake
Juniper

Expires: September 12, 2017 March 12, 2017

Fast Recovery for EVPN DF Election
draft-sajassi-bess-evpn-fast-df-recovery-00

Abstract

Ethernet Virtual Private Network (EVPN) solution [RFC 7432] describes DF election procedures for multi-homing Ethernet Segments. These procedures are enhanced further in [EVPN-DF-Election] by applying Highest Random Weight Algorithm for DF election in order to avoid DF status unnecessarily upon a failure. This draft makes further improvement to DF election procedures in [EVPN-DF-Election] by providing two options for fast DF election upon recovery of the failed link or node associated with the multi-homing Ethernet Segment. This fast DF election is achieved independent of number of EVIs associated with that Ethernet Segment and it is performed via a simple signaling between the recovered PE and each PE in the multi-homing group.

Status of this Memo

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Table of Contents

1 Introduction 4
1.1 Terminology 4
2 Challenges with Existing Solution 4
3 Operation 6
3.1 DF Election Handshake Solution 6
3.1.1 Discovery 6
3.1.2 DF candidates Determination 6
3.1.3 DF Election Handshake 7
3.1.4 Node Insertion 7
3.1.5 BGP Encoding 8
3.1.5.1 DF Election Handshake Request Route 8
3.5.1.2 DF Election Handshake Response Route 9
3.1.6 DF Handshake Scenarios 10
3.1.7 Interoperability 13
3.2 DF Election Synchronization Solution 14
3.2.3 Advantages 15
3.2.4 Interoperability 15
3.2.5 BGP Encoding 15
3.2.6 Note on NTP-based synchronization 16
3.2.7 An example 16
4 Acknowledgement 17
5 Security Considerations 17
6 IANA Considerations 17
7 References 17
7.1 Normative References 17

7.2 Informative References 17
Authors' Addresses 17

1 Introduction

Ethernet Virtual Private Network (EVPN) solution [RFC 7432] is becoming pervasive in data center (DC) applications for Network Virtualization Overlay (NVO) and DC interconnect (DCI) services, and in service provider (SP) applications for next generation virtual private LAN services.

EVPN solution [RFC 7432] describes DF election procedures for multi-homing Ethernet Segments. These procedures are enhanced further in [EVPN-DF-Election] by applying Highest Random Weight Algorithm for DF election in order to avoid DF status change unnecessarily upon a link or node failure associated with the multi-homing Ethernet Segment. This draft makes further improvement to DF election procedures in [EVPN-DF-Election] by providing two options for a fast DF election upon recovery of the failed link or node associated with the multi-homing Ethernet Segment. This DF election is achieved independent of number of EVIs associated with that Ethernet Segment and it is performed via a simple signaling between the recovered PE and each PE in the multi-homing group. The draft presents two signaling options. The first option is based on a bidirectional handshake procedure whereas the second option is based on simple one-way signaling mechanism.

1.1 Terminology

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 RFC 2119 [KEYWORDS].

Provider Edge (PE) : A device that sits in the boundary of Provider and Customer networks and performs encap/decap of data from L2 to L3 and vice-versa.

Designated Forwarder (DF): An PE that is currently forwarding (encapsulating/decapsulating) traffic for a given VLAN in and out of a site.

2 Challenges with Existing Solution

In EVPN technology, multiple PE devices have the ability to encap and decap data belonging to the same VLAN. In certain situations, this may cause L2 duplicates and even loops if there is a momentary overlap of forwarding roles between two or more PE devices, leading to broadcast storms.

EVPN [RFC 7432] currently uses timer based synchronization among PE devices in redundancy group that can result in duplications (and even

loops) because of multiple DFs if the timer is too short or blackholing if the timer is too long.

Using site-of-origin Split Horizon filtering can prevent loops (but not duplicates), however if there are overlapping DFs in two different sites at the same time for the same VLAN, the site identifier will be different upon re-entry of the packet and hence the split horizon check will fail, leading to L2 loops.

The current state of art [EVPN-DF-Election] uses the well known HRW (Highest Random Weight) algorithm to avoid reshuffling of VLANs among PE devices in the redundancy group upon failure/recovery and thus reducing the impact of failure/recovery to VLANs not on the failed/recovered ports. This eliminates loops/duplicates in failure scenarios.

However, upon PE insertion or port bring-up, HRW cannot help as a transfer of DF role need to happen to the newly inserted device/port while the old DF is still active.

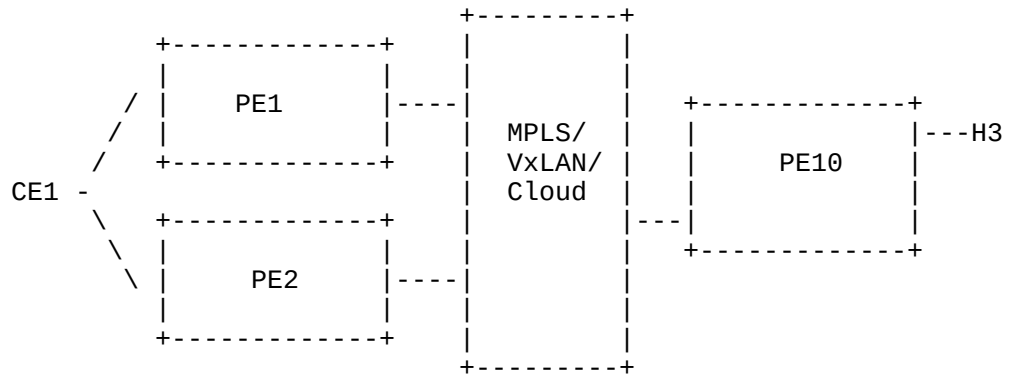


Figure 1: CE1 multi-homed to PE1 and PE2. Potential for duplicate DF.

In the Figure 1, when PE2 is inserted or booted up, PE1 will transfer DF role of some VLANs to PE2 to achieve load balancing. However, because there is no handshake mechanism between PE1 and PE2, duplication of DF roles for a give VLAN is possible. Duplication of DF roles may eventually lead to L2 loops as well as duplication of traffic.

Current state of EVPN art relies on a blackholing timer for transferring the DF role to the newly inserted device. This can cause the following issues:

* Loops/Duplicates if the timer value is too short

* Prolonged Traffic Blackholing if the timer value is too long

This draft is proposing solutions that deterministically eliminates loops/duplicates and at the same time provides fast convergence upon PE/port insertion.

3 Operation

Here we describe two signaling mechanisms between the newly inserted PE and remaining PEs. The signaling is only possible once the newly inserted PE has reliably discovered the other PEs and vice versa. The first option is referred to as DF Election Handshake solution and is described in section 3.1. The second option is referred to as DF Election Synchronization Solution and is described in section 3.2.

3.1 DF Election Handshake Solution

Due to HRW, the handshake will only be one per PE device and independent of EVI/VNI scale. Therefore, this solution is divided into three steps:

Phase 1: Discovery

Phase 2: DF Candidate Determination; HRW

Phase 3: Handshake

Following is the description each step in detail.

3.1.1 Discovery

Each PE needs to have a consistent view of the network including the newly inserted PE.

Newly inserted device PE will advertise it’s Ethernet Segment route and start a flood/wait timer. This timer should be large enough to guarantee the dissemination and receipt of this advertisement by previously inserted PEs.

As the old DF is continuously forwarding traffic while the new PE is running this timer, this timer can be made as long as required without impacting traffic convergence. The timer value can be the BGP session hold time in the worst case to ensure proper discovery.

3.1.2 DF candidates Determination

After the discovery timer has elapsed, each PE would have an imported list of the Ethernet Segment Routes from other PEs. The resultant database will comprise of all the DF candidates on a per ES basis and will be used for DF election. Each PE will independently run the HRW algorithm for all VLANs in a given Ethernet Segment. Since the discovery phase guarantees uniform network view between the participating devices, the HRW VLAN distribution results will be consistent.

3.1.3 DF Election Handshake

The DF Election handshake will be accomplished in the following steps:

- The newly inserted PE will send the DF Request to previously inserted PEs with a new sequence number.

- The previously inserted PE(s) will receive the DF Request, will validate this request as per own discovery state and HRW results.

- The previously inserted PE(s) will program hardware to block the VLANs that must be transferred to the newly inserted PE.

- The previously inserted PE(s) will send DF Response (W/ ACK OR NACK) to the newly inserted PE with the same sequence number that was contained in the DF Request.

- Newly inserted PE will receive DF Response and validate it using the sequence number. It will take action per received DF Response message and will not wait for all previously inserted devices for faster convergence.

- In case of a DF Response ACK, newly inserted PE will program its hardware to assume the DF responsibility.

We don’t need to have a handshake on a per VLAN/EVI basis but rather per pair of PEs in the redundancy group - i.e., if a new PE is added to an existing redundancy group of 3 PE devices, then we need only to have 3 handshakes. This is because the devices already are in sync about which VLANs to give-up/takeover (HRW).

At the end of these three phases, the VLAN DF role transfer would have happened in a deterministic way while ensuring minimum traffic loss. Device recovery and device insertion scenarios are identical in terms of the handshaking procedure. In next section, we describe the procedure details for device insertion.

3.1.4 Node Insertion

Consider the scenario where PE3 is inserted in the network, while PE1 and PE2 are already in stable state. PE3 will send/receive the following flags in the route Type 4:

- DF Request: Upon completing the DF Election, PE3 will send DF Request with a new sequence number. PE1 and PE2 will receive this message and respond with DF Response ACK or NACK with the same sequence number that was generated by PE3.

- DF Response ACK: When PE3 receives DF Response ACK from PE1 with the same sequence number as DF Request, it will take over the DF role for the appropriate VLANS that are being transferred from PE1. When DF Response ACK from PE2 arrives, the rest of the VLANS to be transferred from PE2 to PE3 are then taken over by PE3.

- DF Response NACK: If PE3 receives DF Response NACK from at least one of PE1 or PE2, it will not take over DF role and will start over.

Consider the scenario where two nodes PE3 and PE4 are being inserted at the same time. Both of them will send a DF Request to PE1 and PE2 at around the same time with possibly the same sequence number. When PE1 and PE2 respond with DF Response ACK, it is important to signify exactly whom the response is meant for as it could be for either requester (PE3 or PE4). To remove any ambiguity and false positives, the IP address of the requester MUST be included in the response message to specify who the response is meant for.

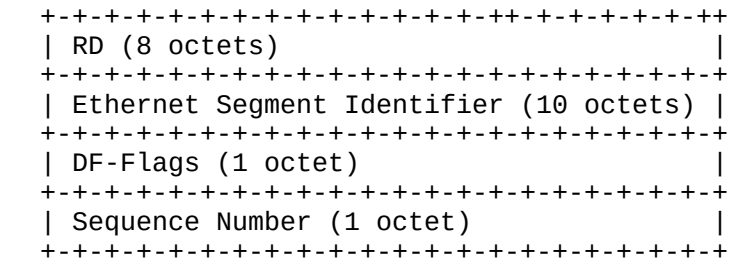
3.1.5 BGP Encoding

The EVPN NLRI comprises of Route Type (1B), Length (1B) and Route Type specific variable encoding. Here we propose the creation of two new EVPN route types:

+ 0x0C - DF Election Handshake Request Route
+ 0x0D - DF Election Handshake Response Route

3.1.5.1 DF Election Handshake Request Route

A DF Election Handshake Request Type NLRI consists of the following:



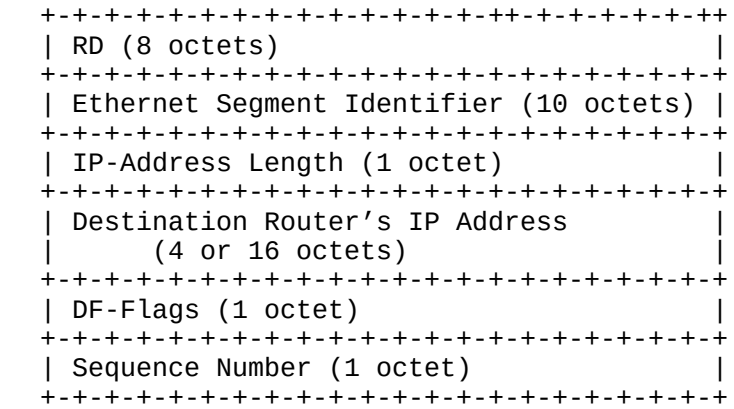
The DF-Flags can have the following values:

DF-INIT : Sent initially upon boot-up; bootstraps the network
DF-REQUEST : Sent to request DF takeover

For the purpose of BGP route key processing, only the Ethernet Segment Identifier is considered to be part of the prefix in the NLRI. The DF-Flag and Sequence number is to be treated as a route attribute as opposed to being part of the route.

3.5.1.2 DF Election Handshake Response Route

A DF Election Handshake Response Type NLRI consists of the following:



The DF-Flags can have the following values:

DF-ACK : Sent to Acknowledge DF-REQUEST
DF-NACK : Sent to Reject DF-Request

For the purpose of BGP route key processing, only the Ethernet Segment Identifier, IP Address Length and Destination Router's IP Address fields are considered to be part of the prefix in the NLRI. The DF-Flag and Sequence number is to be treated as a route attribute as opposed to being part of the route.

3.1.6 DF Handshake Scenarios

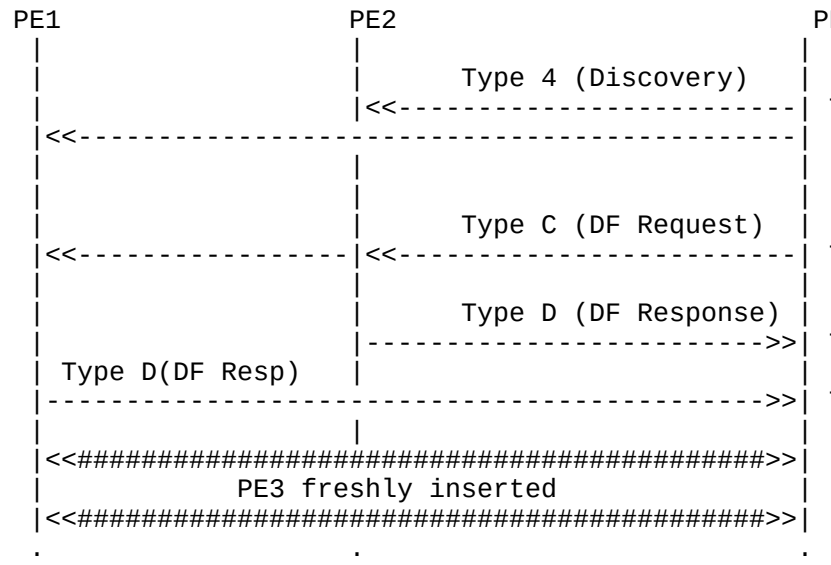
Consider the scenario where PE3 is freshly inserted into the network with PE1 and PE2 in steady state (as shown below). As shown in the sequence diagram below, at time = T0, PE3 will send Type 4 ES route and that will cause PE1 and PE2 to discover PE3.

Post the discovery timer, at time = T1, PE3 will send DF Request containing [ESI, DF-REQ, SEQ1].

PE2 responds via DF Response ACK at time = T2, with the same sequence number SEQ1. [ESI, DF-ACK, PE3, SEQ1]. Note that the sequence number is the same as is contained in the DF Request from PE3. PE3 will receive the DF Response ACK and take over the appropriate VLANs based on HRW only if the sequence number matches.

PE1 responds via DF Response ACK at time = T3, with the same sequence number SEQ1; [ESI, DF-ACK, PE3, SEQ1]. PE3 will receive the DF Response ACK and take over the appropriate VLANs based on HRW only if the sequence number matches.

By the end of the handshake, all appropriate VLANs for the ES are transferred from PE1 and PE2 to PE3 with a single per-ES handshake.

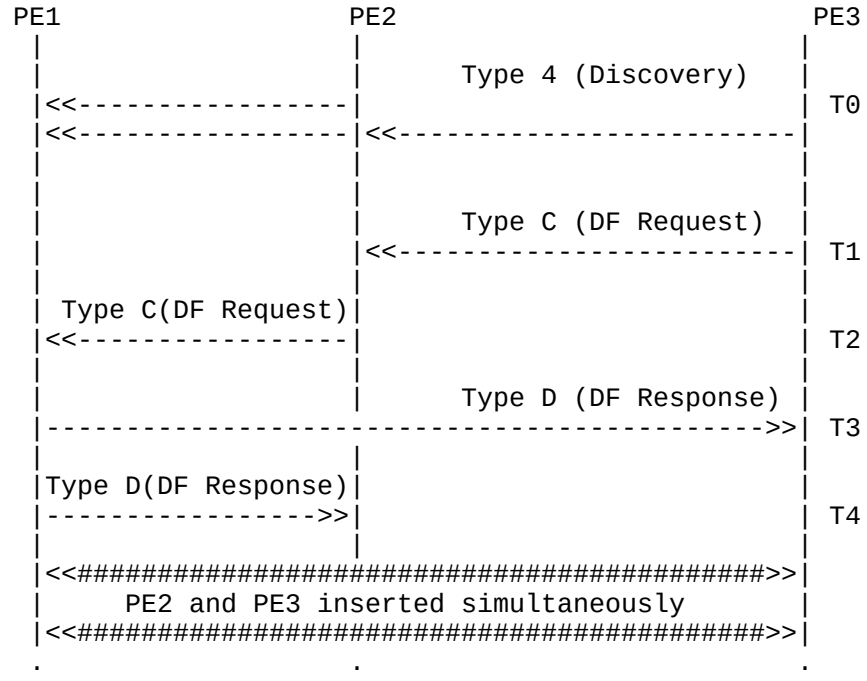


Consider the scenario where PE2 and PE3 are inserted simultaneously in the network where PE1 is in steady state (as shown below). PE2 and PE3 will send the Type 4 ES routes and start the discovery timer. This will cause PE1, PE2 and PE3 to discover each other.

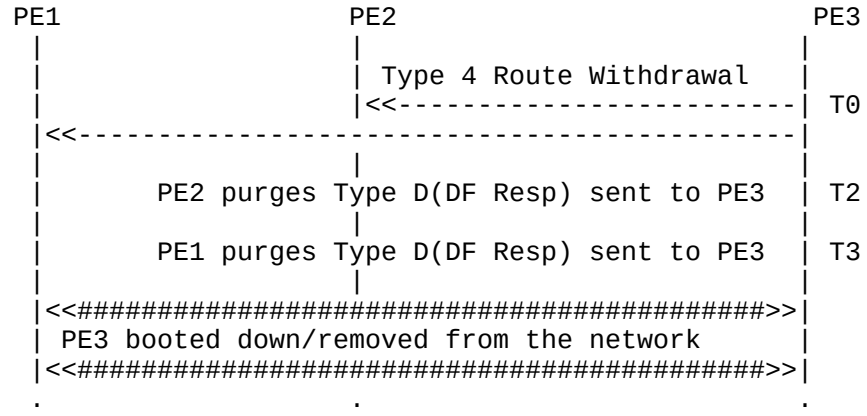
PE2 and PE3 will then simultaneously and separately send DF Request. PE1 will receive these requests and respond to them.

To avoid any ambiguity, PE1 will explicitly specify in the DF Request route the destination for which the DF-ACK is meant for. That is why the responses from PE1 will contain [ES1, DF-ACK, PE2, SEQ] and [ESI, DF-ACK, PE3, SEQ] to specify that the response is meant for PE2 and PE3 respectively.

Upon receiving the Type-D response message, PE2 and PE3 will take over the respective VLANs.



When PE3 is booted down or removed from the network, the routes formerly advertised by PE3 will be withdrawn, including the Type 4 route (as shown below). When PE1 and PE2 process the deletion of PE3's Type 4 route, they will clean up any DF handshake state pertaining to PE3. This means that PE1 and PE2 will withdraw the DF Response routes that they had earlier sent with PE3 as the destination.



3.1.7 Interoperability

Per redundancy group (per ES), for the DF election procedures to be globally convergent and unanimous, it is necessary that all the participating PEs agree on the DF Election algorithm to be used. It is, however, possible that some PEs continue to use the existing modulus based DF election and do not rely on the new handshake/sync procedures. PEs running an old versions of draft/RFC shall simply discard unrecognized new BGP extended communities.

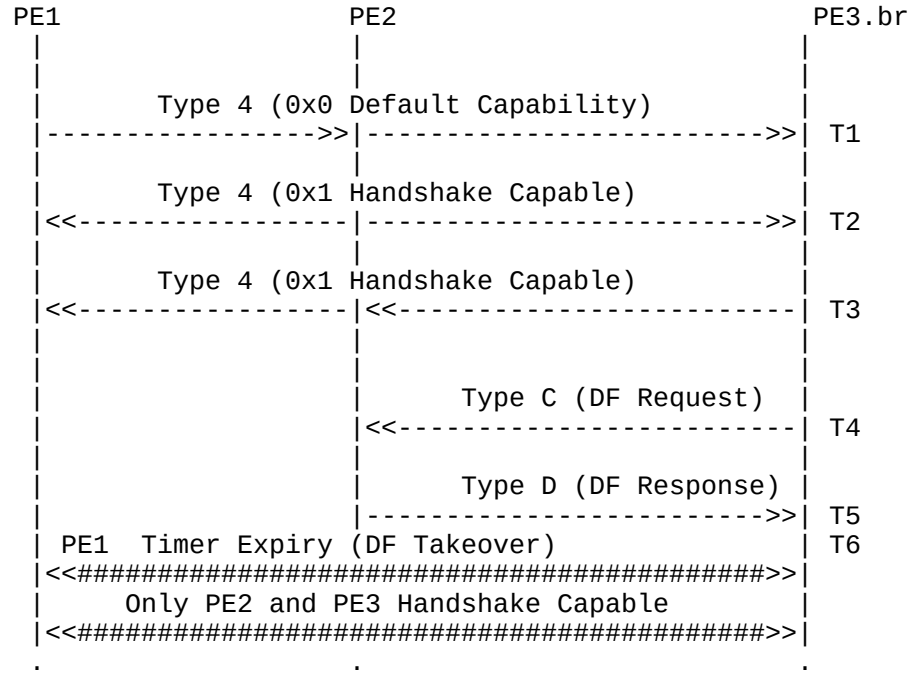
A PE can indicate its willingness to support new DF handshake procedures by signaling DF Election type in the DF Election Extended Community defined in [EVPN-DF] sent along with the Ethernet-Segment Route (Type-4).

Following additional types will be used to indicate the capability:

0x3 : Handshake Based Mechanism
0x4 : Time Sync Based Mechanism

Given that all the PE devices run the same HRW election algorithm, only a subset of them may have the capability of performing the handshake or synchronization mechanism. In such a situation, only the devices that are capable of handshake will partake in handshake and only the devices that are capable of synchronization will partake in sync, rest will default to the timer based mechanism defined in the base RFC.

In the illustration below, PE1, PE2 and PE3 send their respective Type 4 routes indicating their DF capabilities at time T1, T2 and T3 respectively. Only PE2 and PE3 are Handshake capable, hence only PE2 and PE3 partake in DF Handshaking procedure described here at time T4 and T5. PE1 on the other hand, runs the DF election timer and takes over the DF role upon timer expiry at time T6.



3.2 DF Election Synchronization Solution

If all PE devices attached to a given Ethernet Segment are clock-synchronized with each other, then the above handshaking procedures can be simplified and packet loss can be reduced from BGP-propagation time (between recovered PE and the DF PE) to very small time (e.g., milliseconds or less).

The simplified procedure is as follow:

First, the DF election procedure, described in RFC7432, is applied as before.

All PEs attached to a given Ethernet-Segment are clock-synchronized; using a networking protocol for clock synchronization (e.g. NTP, PTP, etc).

Newly inserted device PE or during failure recovery of a PE, that PE communicates the current time to peering partners plus the remaining peering timer time left. This constitute an "endtime" as see from local PE. That "endtime" is called "Service Carving Time" (SCT).

A new BGP Extended Community is advertised along with RT-4 to communicate to other partners the Service Carving Time.

Upon reception of that new BGP Extended Community, partner PEs know exactly its carving time. The notion of skew is introduced to eliminate any potential duplicate traffic or loops. They add a skew (default = -10ms) to the Service Carving Time to enforce this; basically partner PEs must carve first.

To summarize, all peering PEs carve almost simultaneously at the time announced by newly added / recovered PE. The newly added/recovered PE initiates the SCT, carves immediately on peering timer expiry. Other PE receiving RT-4 with a SCT BGP ExtComm, carve shortly before "SCT time".

3.2.3 Advantages

There are multiples advantages of using the approach. Here is a non-exhaustive list:

- A simple uni-directional signaling is all needed

- Backwards-compatible: old versions of draft/RFC shall simply discard unrecognized new SCT BGP ExtComm

- Multiple DF Election algorithms can be supported:
    * RFC7432’s default ordered list ordinal algorithm (modulo)
    * draft-mohanty-bess-evpn-df-election (HRW), etc
- Independent of BGP transmission delay for RT-4
- Solutions is agnostic of the time synchronization mechanisms (e.g. NTP, PTP, ...)

3.2.4 Interoperability

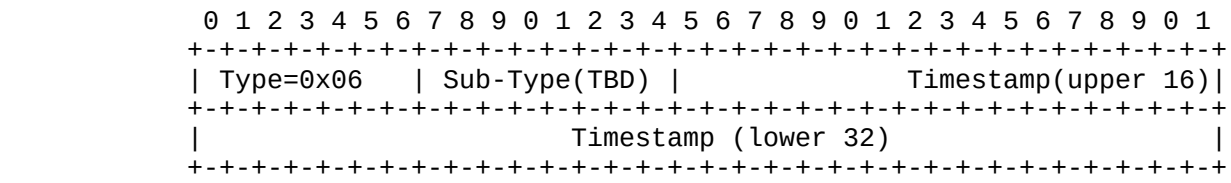
Per redundancy group, for the DF election procedures to be globally convergent and unanimous, it is necessary that all the participating PEs agree on the DF Election algorithm to be used. It is, however, possible that some PEs continue to use the existing modulus based DF election and do not rely on the new SCT BGP extended community. PEs running an baseline DF election mechanism shall simply discard unrecognized new SCT BGP extended community.

A PE can indicate its willingness to support clock-synched carving by signaling the new SCT BGP extended community along with the Ethernet-Segment Route (Type-4).

3.2.5 BGP Encoding

A new BGP extended community needs to be defined to communicate the Service Carving Expected Timestamp for each Ethernet Segment.

A new transitive extended community where the Type field is 0x06, and the Sub-Type is <to be defined> is advertised along with Ethernet Segment route. Timestamp for expected Service carving is encoded as a 8-octet value as follows:



3.2.6 Note on NTP-based synchronization

The 64-bit timestamp used by NTP protocol consists of a 32-bit part for seconds and a 32-bit part for fractional second. Giving a time scale that rolls over every 2^32 seconds (136 years) and a theoretical resolution of 2^32 seconds (233 picoseconds). The recommendation is to keep the top 32 bits and carry lower MSB 16 bits of fractional second.

3.2.7 An example

Let’s take figure 1 as an example where initially PE2 had failed and PE1 had taken over.

Based on RFC-7432:

- Initial state: PE1 is in steady-state, PE2 is recovering
- PE2 recovers at (absolute) time t=99
- PE2 advertises RT-4 (sent at t=100) to partner PE1.
- PE2, it starts its 3sec peering timer as per RFC7432
- PE1 carves immediately on RT-4 reception. PE2 carves at time t=103.

With following procedure, there is a high chance to generate a traffic black hole or traffic loop. The peering timer value has a direct effect of this behavior. A short peering timer may generate loop whereas a long peering timer provide a prolong blackout.

Based on the SCT approach:

- Initial state: PE1 is in steady-state, PE2 is recovering

- PE2 recovers at (absolute) time t=99
- PE2 advertises RT-4 (sent at t=100) with target SCT value t=103 to partner PE1
- PE2 starts its 3sec peering timer as per RFC7432
- Both PE1 and PE2 carves at (absolute) time t=103; In fact, PE1 should carve slightly before PE2 (skew).

Using SCT approach, the effect of the peering timer is gone. Also, the BGP RT-4 transmission delay (from PE2 to PE1) becomes a no-op.

4 Acknowledgement Authors would like to acknowledge helpful comments and contributions of Satya Mohanty and Luc Andre Burdet.

5 Security Considerations

The mechanisms in this document use EVPN control plane as defined in [RFC7432]. Security considerations described in [RFC7432] are equally applicable. This document uses MPLS and IP-based tunnel technologies to support data plane transport. Security considerations described in [R7432] and in [ietf-evpn-overlay] are equally applicable.

6 IANA Considerations

Allocation of Extended Community Type and Sub-Type for EVPN.

7 References

7.1 Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC7432] Sajassi et al., "BGP MPLS Based Ethernet VPN", February, 2015.

7.2 Informative References

[EVPN-DF]Key et al., "A new Designated Forwarder Election for the EVPN", draft-ietf-bess-evpn-df-election-01, work in progress, April 2017.

Authors' Addresses

Ali Sajassi
Cisco
Email: sajassi@cisco.com

Gaurav Badoni
Cisco
Email: gbadoni@cisco.com

Patrice Brissette
Cisco
Email: pbrisset@cisco.com

Dhananjaya Rao
Cisco
Email: dhrao@cisco.com

John Drake
Juniper
Email: jdrake@juniper.net

BESS
Internet-Draft
Intended status: Standards Track
Expires: September 14, 2017

Z. Zhang
Juniper Networks
K. Patel
Arrcus
I. Wijnands
Cisco Systems
A. Gulko
Thomson Reuters
March 13, 2017

BGP Based Multicast
draft-zzhang-bess-bgp-multicast-01

Abstract

This document specifies a BGP address family and related procedures that allow BGP to be used for setting up multicast distribution trees. This document also specifies procedures that enable BGP to be used for multicast source discovery, and for showing interest in receiving particular multicast flows. Taken together, these procedures allow BGP to be used as a replacement for other multicast routing protocols, such as PIM or mLDP. The BGP procedures specified here are based on the BGP multicast procedures that were originally designed for use by providers of Multicast Virtual Private Network service.

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

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 http://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 September 14, 2017.

Table of Contents

1. Introduction 3
1.1. Motivation 3
1.1.1. Native/unlabeled Multicast 3
1.1.2. Labeled Multicast 4
1.2. Overview 4
1.2.1. (x,g) Multicast 4
1.2.1.1. Source Discovery for ASM 5
1.2.1.2. ASM Shared-tree-only Mode 6
1.2.1.3. Integration with BGP-MVPN 6
1.2.2. BGP Inband Signaling for mLDP Tunnel 7
1.2.3. BGP Sessions 7
1.2.4. LAN and Parallel Links 8
1.2.5. Transition 9
2. Specification 9
2.1. BGP NLRIs and Attributes 9
2.1.1. S-PMSI A-D Route 10
2.1.2. Leaf A-D Route 11
2.1.3. Source Active A-D Route 12
2.1.4. S-PMSI A-D Route for C-multicast mLDP 12
2.1.5. Session Address Extended Community 12
2.2. Procedures 13
2.2.1. Source Discovery for ASM 13
2.2.2. Originating Tree Join Routes 13
2.2.2.1. (x,g) Multicast Tree 13
2.2.2.2. BGP Inband Signaling for mLDP Tunnel 14
2.2.3. Receiving Tree Join Routes 15
2.2.4. Withdrawl of Tree Join Routes 15
2.2.5. LAN procedures for (x,g) Unidirectional Tree 15
2.2.5.1. Originating S-PMSI A-D Routes 15

2.2.5.2. Receiving S-PMSI A-D Routes 16
2.2.6. Distributing Label for Upstream Traffic for Bidirectional Tree/Tunnel 17
3. Security Considerations 17
4. Acknowledgements 17
5. References 17
5.1. Normative References 17
5.2. Informative References 19
Authors' Addresses 19

1. Introduction

1.1. Motivation

This section provides some motivation for BGP signaling for native and labeld multicast. One target deployment would be a Data Center that requires multicast but uses BGP as its only routing protocol [RFC7938]. In such a deployment, it would be desirable to support multicast by extending the deployed routing protocol, without requiring the deployment of tree building protocols such as PIM, mLDP, RSVP-TE P2MP, and without requiring an IGP.

Additionally, compared to PIM, BGP based signaling has several advantage as described in the following section, and may be desired in non-DC deployment scenarios as well.

1.1.1. Native/unlabeled Multicast

Protocol Independent Multicast (PIM) has been the prevailing multicast protocol for many years. Despite its success, it has two drawbacks:

- The ASM model, which is prevalent, introduces complexity in the following areas: source discovery procedures, need for Rendezvous Points (RPs) and group-to-RP mappings, need to switch between RP-rooted trees and source-rooted trees, etc.

- Periodical protocol state refreshes due to soft state nature.

While PIM-SSM removes the complexity of PIM-ASM, it requires that multicast sources are known apriori. There have not been a good way of discovering sources, so its deployment has been limited. PIM-Port (PIM over Reliable Transport) solves the soft state issue, though its deployment has also been limited for two reasons:

- It does not remove the ASM complexities.

- o In many of the scenarios where reliable transport is deemed important, BGP-based multicast (e.g. BGP-MVPN) has been used instead of PORT.

Partly because of the above mentioned problems, some Data Center operators have been avoiding deploying multicast in their networks.

BGP-MVPN [RFC6514] uses BGP to signal VPN customer multicast state over provider networks. It removes the above mentioned problems from the SP environment, and the deployment experiences have been encouraging. While RFC 6514 makes it possible for an SP to provide MVPN service without running PIM on its backbone, that RFC still assumes that PIM (or mLDP) runs on the PE-CE links. [draft-ietf-bess-mvpn-pe-ce] adapts the concept of BGP-MVPN to PE-CE links so that the use of PIM on the PE-CE links can be eliminated (though the PIM-ASM complexities still remains in the customer network), and this document extends it further to general topologies, so that they can be run on any router, as a replacement for PIM or mLDP.

With that, PIM can be completely eliminated from the network. PIM soft state is replaced by BGP hard state. For ASM, source specific trees are set up directly after simpler source discovery (data driven on FHRs and control driven elsewhere), all based on BGP. All the complexities related to source discovery and shared/source tree switch are also eliminated. Additionally, the trees can be setup with MPLS labels, with just minor enhancements in the signaling.

1.1.2. Labeled Multicast

There could be two forms of labeled multicast signaled by BGP. The first one is labeled (x,g) multicast where 'x' stands for either 's' or '*'. Basically, it is for BGP-signaled multicast tree as described in previous section but with labels. The second one is for mLDP tunnels with BGP signaling in part or whole through a BGP domain.

For both cases, BGP is used because other label distribution mechanisms like mLDP may not be desired by some operators. For example, a DC operator may prefer to have a BGP-only deployment.

1.2. Overview

1.2.1. (x,g) Multicast

PIM-like functionality is provided, using BGP-based join/prune signaling and BGP-based source discovery for ASM. The BGP-based join signaling supports both labeled multicast and IP multicast.

The same RPF procedures as in PIM are used for each router to determine the RPF neighbor for a particular source or RPA (in case of Bidirectional Tree). Except in the Bidirectional Tree case and a special case described in Section 1.2.1.2, no (*,G) join is used - LHR routers discover the sources for ASM and then join towards the sources directly. Data driven mechanisms like PIM Assert is replaced by control driven mechanisms (Section 1.2.4).

The joins are carried in BGP Updates with C-MCAST SAFI defined in [draft-ietf-bess-mvpn-pe-ce] and S-PMSI/Leaf A-D routes defined in this document. The updates are targeted at the upstream neighbor by use of Route Targets. [Note - earlier version of this draft uses C-multicast route to send joins. We’re now switching to S-PMSI/Leaf routes for three reasons. a) when the routes go through RRs, we have to distinguish different routes based on upstream router and downstream router. This leads to Leaf routes. b) for labeled bidirectional trees, we need to signal "upstream fec". S-PMSI suits this very well. c) we may want to allow the option of setting up trees from the roots instead of from the leaves. S-PMSI suits that very well.]

If the BGP updates carry labels (via Tunnel Encapsulation Attribute [I-D.ietf-idr-tunnel-encaps]), then (s,g) multicast traffic can use the labels. This is very similar to mLDP Inband Signaling [RFC6826], except that there are no corresponding "mLDP tunnels" for the PIM trees. Similar to mLDP, labeled traffic on transit LANs are point to point. Of course, traffic sent to receivers on a LAN by a LHR is native multicast.

For labeled bidirectional (*,g) trees, downstream traffic (away from the RPA) can be forwarded as in the (s,g) case. For upstream traffic (towards RPA), the upstream neighbor needs to advertise a label for its downstream neighbors. The same label that the upstream neighbor advertises to its upstream is the same one that it advertises to its downstreams, using an S-PMSI A-D route.

1.2.1.1. Source Discovery for ASM

This document does not support ASM via shared trees (aka RP Tree, or RPT) with one exception discussed in the next section. Instead, FHRs, LHRs, and optionally RRs work together to propagate/discover source information via control plane and LHRs join source specific Shortest Path Trees (SPT) directly.

A FHR originates Source Active A-D routes upon discovering sources for particular flows and advertise them to its peers. It is desired that the SA routes only reach LHRs that are interested in receiving the traffic. To achieve that, the SA routes carry an IPv4 or IPv6

address specific Route Target. The Global Administrator field is set the group address of the flow, and the Local Administrator field is set to 0. An LHR advertises Route Target Membership routes, with the Route Target field in the NLRI set according to the groups it wants to receive traffic for, as how a FHR encode the Route Target in its Source Active routes. The propagation of the SA routes is subject to cooperative export filtering as specified in [RFC4684] and referred to as RTC mechanism in this document. That way, the LHR only receives Source Active routes for groups that it is interested in.

Typically, a set of RRs are used and they maintains all Source Active routes but only distribute to interested LHRs on demand (upon receiving corresponding Route Target Memberhip routes, which are triggered on LHRs when they receive IGMP/MLD membership routes). The rest of the document assumes that RRs are used, even though that is not required.

1.2.1.2. ASM Shared-tree-only Mode

It may be desired that only a shared tree is used to distribute all traffic for a particular ASM group from its RP to all LHRs, as described in Section 4.1 "PIM Shared Tree Forwarding" of [RFC7438]. This will significantly cut down the number of trees and works out very well in certain deployment scenarios. For example, all the sources could be connected to the RP, or clustered close the to RP. In the latter case, either the path from FHRs to the RP do not intersect the shared tree so natvie forwarding can be used between the FHRs and the RP, or other means outside of this document could be used to forward traffic from FHRs to the RP.

For native forwading from FHRs to the RP, SA routes may be used to announce the sources so that the RP can join source specific trees to pull traffic, but the group specific Route Target is not needed. The LHRs do not advertise the group specific Route Target Membership routes as they do not need the SA routes.

To establish the shared tree, (*,g) Leaf A-D routes are used as in the bidirectional tree case, though no forwarding state is established to forward traffic from downstream neighbors.

1.2.1.3. Integration with BGP-MVPN

For each VPN, the Source Active routes distribution in that VPN do not have to invlove PEs at all unless there are sources/receivers directly connected to some PEs and they are independent of MVPN SA routes. For example, FHRs and LHRs establish BGP sessions with RRs of that particular VPN for the purpose of SA distribution.

After source discovery, BGP multicast signaling is done from LHRs towards the sources. When the signaling reaches an egress PE, BGP-MVPN signaling takes over, as if a PIM (s,g) join/prune was received on the PE-CE interface. When the BGP-MVPN signaling reaches the ingress PE, BGP multicast signaling as specified in this document takes over, similar to how BGP-MVPN triggers PIM (s,g) join/prune on PE-CE interfaces.

1.2.2. BGP Inband Signaling for mLDP Tunnel

Part of an (or the whole) mLDP tunnel can also be signaled via BGP and seamlessly integrated with the rest of mLDP tunnel signaled natively via mLDP. All the procedures are similar to mLDP except that the signaling is done via BGP. The mLDP FEC is encoded as the BGP NLRI, with C-MCAST SAFI and S-PMSI/Leaf A-D Routes for C-multicast mLDP defined in this document. The Leaf A-D routes correspond to mLDP Label Mapping messages, and the S-PMSI A-D routes are used to signal upstream FEC for MP2MP mLDP tunnels, similar to the bidirection (*,g) case.

1.2.3. BGP Sessions

As specified in [draft-ietf-bess-mvpn-pe-ce-00], in order for two BGP speakers to exchange C-MCAST NLRI, they must use BGP Capabilities Advertisement [RFC5492] to ensure that they both are capable of properly processing the C-MCAST NLRI. This is done as specified in [RFC4760], by using a capability code 1 (multiprotocol BGP) with an AFI of IPv4 (1) or IPv6 (2) and a SAFI of C-MCAST with a value to be assigned by IANA.

How the BGP peer sessions are provisioned, whether EBGP or IBGP, whether statically, automatically (e.g., based on IGP neighbor discovery), or programmably via an external controller, is outside the scope of this document.

In case of IBGP, it could be that every router peering with Route Reflectors, or hop by hop IBGP sessions could be used to exchange C-MCAST NLRIs for joins. In the latter case, unless desired otherwise for reasons outside of the scope of this document, the hop by hop IBGP sessions SHOULD only be used to exchange C-MCAST NLRIs.

When multihop BGP is used, a router advertises its local interface addresses, for the same purposes that the Address List TLV in LDP serves. This is achieved by advertising the interface address as host prefixes with IPv4/v6 Adderss Specific ECs corresponding to the router’s local addresses used for its BGP sessions (Section 2.1.5).

Because the BGP Capability Advertisement is only between two peers, when the sessions are only via RRs, a router needs another way to determine if its neighbor is capable of signaling multicast via BGP. The interface address advertisement can be used for that purpose - the inclusion of a Session Address EC indicates that the BGP speaker identified in the EC supports the C-Multicast NLRI.

FHRs and LHRs may also establish BGP sessions to some Route Reflectors for source discovery purpose (Section 1.2.1.1).

With the traditional PIM, the FHRs and LHRs refer to the PIM DRs on the source or receiver networks. With BGP based multicast, PIM may not be running at all, and the FHRs and LHRs refer to the IGMP/MLD queriers, or the DF elected per [I-D.wijnands-bier-mld-lan-election]. Alternatively, if it is known that a network only has senders then no IGMP/MLD or DF election is needed - any router may generate SA routes. That will not cause any issue other than redundnant SA routes being originated.

1.2.4. LAN and Parallel Links

There could be parallel links between two BGP peers. A single multi-hop session, whether IBGP or EBGP, between loopback addresses may be used. Except for LAN interfaces in case of unlabeled (x,g) unidirectional trees (note that transit LAN interface is not supported for BGP signaled (*,g) bidirectional tree and for mLDP tunnels, traffic on transit LAN is point to point between neighbors), any link between the two peers can be automatically used by a downstream peer to receive traffic from the upstream peer, and it is for the upstream peer to decide which link to use. If one of the links goes down, the upstream peer switches to a different link and there is no change needed on the downstream peer.

For unlabeled (x,g) unidirectional trees, the upstream peer MAY prefer LAN interfaces to send traffic, since multiple downstream peers may be reached simultaneously, or it may make a decision based on local policy, e.g., for load balancing purpose. Because different downstream peers might choose different upstream peers for RPF, when an upstream peer decides to use a LAN interface to send traffic, it originates an S-PMSI A-D route indicating that one or more LAN interface will be used. The route carries Route Targets specific to the LANs so that all the peers on the LANs import the route. If more than one router originate the route specifying the same LAN for the same (s,g) or (*,g) flow, then assert procedure based on the S-PMSI A-D routes happens and assert losers will stop sending traffic to the LAN.

1.2.5. Transition

A network currently running PIM can be incrementally transitioned to BGP based multicast. At any time, a router supporting BGP based multicast can use PIM with some neighbors (upstream or downstream) and BGP with some other neighbors. PIM and BGP MUST not be used simultaneously between two neighbors for multicast purpose, and routers connected to the same LAN MUST be transitioned during the same maintenance window.

In case of PIM-SSM, any router can be transitioned at any time (except on a LAN all routers must be transitioned together). It may receive source tree joins from a mixed set of BGP and PIM downstream neighbors and send source tree joins to its upstream neighbor using either PIM or BGP signaling.

In case of PIM-ASM, the RPs are first upgraded to support BGP based multicast. They learn sources either via PIM procedures from PIM FHRs, or via Source Active A-D routes from BGP FHRs. In the former case, the RPs can originate proxy Source Active A-D routes. There may be a mixed set of RPs/RRs - some capable of both traditional PIM RP functionalities while some only redistribute SA routes.

Then any routers can be transitioned incrementally. A transitioned LHR router will pull Source Active A-D routes from the RPs/RRs when they receive IGMP/MLD (*,G) joins for ASM groups, and may send either PIM (s,g) joins or BGP Source Tree Join routes. A transitioned transit router may receive (*,g) PIM joins but only send source tree joins after pulling Source Active A-D routes from RPs/RRs.

Similarly, a network currently running mLDP can be incrementally transitioned to BGP signaling. Without the complication of ASM, any router can be transitioned at any time, even without the restriction of coordinated transition on a LAN. It may receive mixed mLDP label mapping or BGP updates from different downstream neighbors, and may exchange either mLDP label mapping or BGP updates with its upstream neighbors, depending on if the neighbor is using BGP based signaling or not.

2. Specification

2.1. BGP NLRIs and Attributes

The C-MCAST SAFI defined in [I-D.ietf-bess-mvpn-pe-ce] is used, but new route types are used as defined in this document.

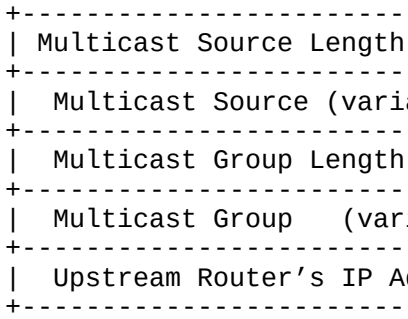
```
          3 -  S-PMSI A-D Route for (x,g)
          4 -  Leaf A-D Route
          5 -  Source Active A-D Route
       0x43 -  S-PMSI A-D Route for C-multicast mLDP
```

Except for the Source Active A-D routes, the routes are to be consumed by targeted upstream/downstream neighbors, and are not propagated further. This can be achieved by outbound filtering based on the RTs that lead to the importation of the routes.

The Type-3/4 routes MAY carry a Tunnel Encapsulation Attribute (TEA) [I-D.ietf-idr-tunnel-encaps]. The Type-0x43 route MUST carry a TEA. When used for mLDP, the Type-4 route MUST carry a TEA. Only the MPLS tunnel type for the TEA is considered. Others are outside the scope of this document.

2.1.1. S-PMSI A-D Route

Similar to defined in RFC 6514, an S-PMSI A-D Route Type specific C-MCAST NLRI consists of the following, though it does not have an RD:



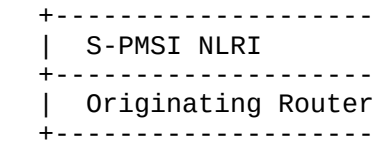
If the Multicast Source (or Group) field contains an IPv4 address, then the value of the Multicast Source (or Group) Length field is 32. If the Multicast Source (or Group) field contains an IPv6 address, then the value of the Multicast Source (or Group) Length field is 128.

Usage of other values of the Multicast Source Length and Multicast Group Length fields is outside the scope of this document.

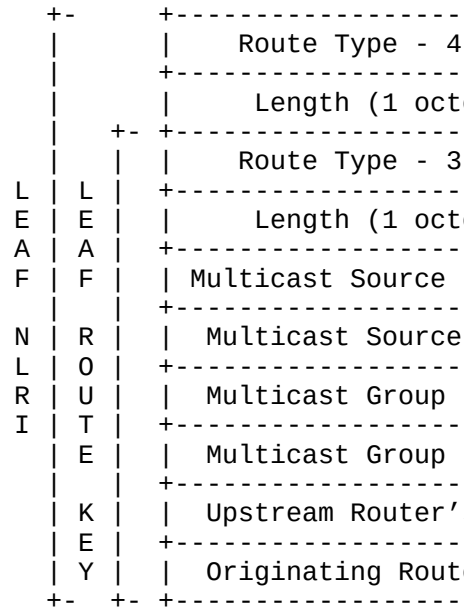
There are two usages for S-PMSI A-D route. They’re described in Section 2.2.5 and Section 2.2.6 respectively.

2.1.2. Leaf A-D Route

Similar to the Leaf A-D route in [RFC6514], a C-MCAST Leaf A-D route's route key includes the corresponding S-PMSI NLRI, plus the Originating Router's IP Addr. The difference is that there is no RD.



For example, the entire NLRI of a Leaf A-D route for (x,g) tree is as following:

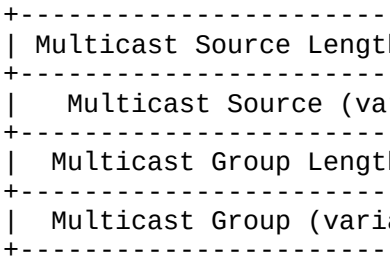


Even though the C-MCAST Leaf A-D route is unsolicited, unlike the Leaf A-D route for GTM in [RFC7524], it is encoded as if a corresponding S-PMSI A-D route had been received.

When used for signaling mLDP tunnels, even though the Leaf A-D route is unsolicited, unlike the "Route-type 0x44 Leaf A-D route for C-multicast mLDP" as in [RFC7441], it is Route-type 4 and encoded as if a corresponding S-PMSI A-D route had been received.

2.1.3. Source Active A-D Route

Similar to defined in RFC 6514, a Source Active A-D Route Type specific MCAST NLRI consists of the following:



The definition of the source/length and group/length fields are the same as in the S-PMSI A-D routes.

Usage of Source Active A-D routes is described in Section 1.2.1.1.

2.1.4. S-PMSI A-D Route for C-multicast mLDP

The route is used to signal upstream FEC for an MP2MP mLDP tunnel. The route key include the mLDP FEC and the Upstream Router's IP Address field. The encoding is similar to the same route in [RFC7441], though there is no RD.

2.1.5. Session Address Extended Community

For two BGP speakers to determine if they are directly connected, each will advertise their local interface addresses, with an Session Address Extended Community. This is an Address Specific EC, with the Global Admin Field set to the local address used for its multihop sessions and the Local Admin Field set to the prefix length corresponding to the interface's network mask.

For example, if a router has two interfaces with address 10.10.10.1/24 and 10.12.0.1/16 respectively (notice the different network mask), and a loopback address 11.11.11.1/32 that is used for BGP sessions, then it will advertise prefix 10.10.10.1/32 with a Session Address EC 11.11.11.1:24 and 10.12.0.1/32 with a Session Address EC 11.11.11.1:16. If it also uses another loopback address 11.11.11.11/32 for other BGP sessions, then the routes will additionally carry Session Address EC 11.11.11.11:24 and 11.11.11.11:16 respectively.

This achieves what the Address List TLV in LDP Address Messages achieves, and can also be used to indicate that a router supports the BGP multicast signaling procedures specified in this document.

Only those interface addresses that will be used as resolved nexthops in the RIB need to be advertised with the Session Address EC. For example, the RPF lookup may say that the resolved nexthop address is A1, so the router needs to find out the corresponding BGP speaker with address A1 through the (interface address, session address) mapping built according to the interface address NLRI with the Session Address EC. For comparison with LDP, this is done via the (interface address, session address) mapping that is built by the LDP Address Messages.

2.2. Procedures

2.2.1. Source Discovery for ASM

When a FHR first receives a multicast packet addressed to an ASM group, it originates a Source Active route. It carries a IP/IPv6 Address Specific RT, with the Global Admin Field set to the group address and the Local Admin Field set to 0. The route is advertised to its peers, who will re-advertise further based on the RTC mechanisms. Note that typically the route is advertised only to the RRs.

The FHRs withdraws the Source Active route after a certain amount of time since it last received a packet of an (s,g) flow. The amount of time to wait is a local matter.

2.2.2. Originating Tree Join Routes

Note that in this document, tree join routes are S-PMSI/Leaf A-D routes.

2.2.2.1. (x,g) Multicast Tree

When a router learns from IGMP/MLD or a downstream PIM/BGP peer that it needs to join a particular (s,g) tree, it determines the RPF nexthop address wrt the source, following the same RPF procedures as defined for PIM. It further finds the BGP router that advertised the nexthop address as one of its local addresses.

If the RPF neighbor supports C-MCAST SAFI, this router originates a Leaf A-D route. Although it is unsolicited, it is constructed as if there was a corresponding S-PMSI A-D route. The Upstream Router’s IP Address field is set to the RPF neighbor’s session address (learnt via the EC attached to the host route for the RPF nexthop address).

An Address Specific RT corresponding to the session address is attached to the route, with the Global Administrative Field set to the session address and the local administrative field set to 0.

Similarly, when a router learns that it needs to join a bi-directional tree for a particular group, it determines the RPF neighbor wrt the RPA. If the neighbor supports C-MCAST SAFI, it originates a Leaf A-D Route and advertises the route to the RPF neighbor (in case of EBGP or hop-by-hop IBGP), or one or more RRs.

When a router first learns that it needs to receive traffic for an ASM group, either because of a local (*,g) IGMP/MLD report or a downstream PIM (*,g) join, it originates a RTC route with the NLRI's AS field set to its AS number and the Route Target field set to an address based RT, with the Global Administrator field set to group address and the Local Administrator field set to 0. The route is advertised to its peers (most practically some RRs), so that the router can receive matching Source Active A-D routes. Upon the receiving of the Source Active A-D routes, the router originates Leaf A-D routes as described above, as long as it still needs to receive traffic for the flows (i.e., the corresponding IGMP/MLD membership exists or join from downstream PIM/BGP neighbor exists).

When a Leaf A-D route is originated by this router, it sets up corresponding forwarding state such that the expected incoming interface list includes all non-LAN interfaces directly connecting to the upstream neighbor. LAN interfaces are added upon receiving corresponding S-PMSI A-D route (Section 2.2.5.2). If the upstream neighbor is not directly connected, tunnels may be used - details to be included in future revisions.

When the upstream nbr changes, the previously advertised Leaf A-D route is withdrawn. If there is a new upstream neighbor, a new Leaf A-D route is originated, corresponding to the new neighbor. Because NLRIs are different for the old and new Leaf A-D routes, make-before-break can be achieved, so can MoFRR [RFC7431].

2.2.2.2. BGP Inband Signaling for mLDP Tunnel

The same mLDP procedures as defined in [RFC6388] are followed, except that where a label mapping message is sent in [RFC6388], a Leaf A-D route is sent if the the upstream neighbor supports BGP based signaling.

2.2.3. Receiving Tree Join Routes

A router (auto-)configures Import RTs matching itself so that it can import tree join routes from their peers. Note that in this document, tree join routes are S-PMSI/Leaf A-D routes.

When a router receives a tree join route and imports it, it determines if it needs to originate its own corresponding route and advertise further upstream wrt the source/RPA or mLDP tunnel root. If itself is the FHR or is on the RPL or is the tunnel root, then it does not need to. Otherwise the procedures in Section 2.2.2 are followed.

Additionally, the router sets up its corresponding forwarding state such that traffic will be sent to the downstream neighbor, and received from the downstream neighbor in case of birectional tree/ tunnel. If the downstream neighbor is not directly connected, tunnels may be used - details to be included in future revisions.

2.2.4. Withdrawl of Tree Join Routes

For a particular tree or tunnel, if a downstream neighbor withdraw its Leaf A-D route, the neighbor is removed from the corresponding forwarding state. If all downstream neighbors withdraw their tree join routes and this router no longer has local receivers, it withdraws the tree join routes that it previously originated.

As mentioned earlier, when the upstream neighbor changes, the previously advertised Leaf A-D route is also withdrawn. The corresponding incoming interfaces are also removed from the corresponding forwarding state.

2.2.5. LAN procedures for (x,g) Unidirectional Tree

For a unidirectional (x,g) multicast tree, if there is a LAN interface connecting to the downstream neighbor, it MAY be preferred over non-LAN interfaces, but an S-PMSI A-D route MUST be originated to facilitate the analog of the Assert process (Section 2.2.5.1).

2.2.5.1. Originating S-PMSI A-D Routes

If this router chooses to use a LAN interface to send traffic to its neighbors for a particular (s,g) or (*,g) flow, it MUST announce that by originating a corresponding S-PMSI A-D route. The Tunnel Type in the PMSI Tunnel Attribute (PTA) is set to 0 (no tunnel information Present). The LAN interface is identified by an IP address specific RT, with the Global Administrative Field set to the LAN interface’s address prefix and the Local Administrative Field set to the prefix

length. The RT also serves the purpose of restricting the importing of the route by all routers on the LAN. An operator MUST ensure that RTs encoded as above are not used for other purposes. Practically that should not be unreasonable.

If multiple LAN interfaces are to be used (to reach different sets of neighbors), then the route will include multiple RTs, one for each used LAN interface as described above.

The S-PMSI A-D routes may also be used to announce tunnels that could be used to send traffic to downstream neighbors that are not directly connected. Details may be added in future revisions.

2.2.5.2. Receiving S-PMSI A-D Routes

A router (auto-)configures an Import RT for each of its LAN interfaces over which BGP is used for multicast signaling. The construction of the RT is described in the previous section.

When a router R1 imports an S-PMSI A-D route for flow (x,g) from router R2, R1 checks to see if it also originating an S-PMSI A-D route with the same NLRI except the Upstream Router’s IP Address field. When a router R1 originates an S-PMSI A-D route, it checks to see if it also has installed an S-PMSI A-D route, from some other router R2, with the same NLRI except the Upstream Router’s IP Address field. In either case, R1 checks to see if the two routes have an RT in common and the RT is encoded as in Section 2.2.5.1. If so, then there is a LAN attached to both R1 and R2, and both routers are prepared to send (S,G) traffic onto that LAN. This kicks off the assert procedure to elect a winner - the one with the highest Upstream Router’s IP Address in the NLRI wins. An assert loser will not include the corresponding LAN interface in its outgoing interface list, but it keeps the S-PMSI A-D route that it originates.

If this router does not have a matching S-PMSI route of its own with some common RTs, and the originator of the received S-PMSI route is a chosen upstream neighbor for the corresponding flow, then this router updates its forwarding state to include the LAN interface in the incoming interface list. When the last S-PMSI route with a RT matching the LAN is withdrawn later, the LAN interface is removed from the incoming interface list.

Note that a downstream router on the LAN does not participate in the assert procedure. It adds/keeps the LAN interface in the expected incoming interfaces as long as its chosen upstream peer originates the S-PMSI AD route. It does not switch to the assert winner as its upstream. An assert loser MAY keep sending joins upstream based on

local policy even if it has no other downstream neighbors (this could be used for fast switch over in case the assert winner would fail).

2.2.6. Distributing Label for Upstream Traffic for Bidirectional Tree/ Tunnel

For MP2MP mLDP tunnels or labeled (*,g) bidirectional trees, an upstream router needs to advertise a label to all its downstream neighbors so that the downstream neighbors can send traffic to itself.

For MP2MP mLDP tunnels, the same procedures for mLDP are followed except that instead of MP2MP-U Label Mapping messages, S-PMSI A-D Routes for C-Multicast mLDP are used.

For labeled (*,g) bidirectional trees, for a Leaf A-D route received from a downstream neighbor, a corresponding S-PMSI A-D route is sent back to the downstream router.

In both cases, a single S-PMSI A-D route is originated for each tree from this router, but with multiple RTs (one for each downstream neighbor on the tree). A TEA specifies a label allocated by the upstream router for its downstream neighbors to send traffic with. Note that this is still a "downstream allocated" label (the upstream router is "downstream" from traffic direction point of view).

The S-PMSI routes do not carry a PTA, unless a P2MP tunnel is used to reach downstream neighbors. Such use case is out of scope of this document for now and may be specified in the future.

3. Security Considerations

This document does not introduce new security risks?

4. Acknowledgements

The authors thank Marco Rodrigues and Lenny Giuliano for their initial idea/ask of using BGP for multicast signaling beyond MVPN. We also thank Eric Rosen for his questions, suggestions, and help finding solutions to some issues.

5. References

5.1. Normative References

[I-D.ietf-bess-mvpn-pe-ce]
Patel, K., Rosen, E., and Y. Rekhter, "BGP as an MVPN PE-CE Protocol", draft-ietf-bess-mvpn-pe-ce-01 (work in progress), October 2015.

[I-D.ietf-idr-tunnel-encaps]
Rosen, E., Patel, K., and G. Velde, "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-03 (work in progress), November 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>.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <http://www.rfc-editor.org/info/rfc4601>.

[RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, DOI 10.17487/RFC4684, November 2006, <http://www.rfc-editor.org/info/rfc4684>.

[RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <http://www.rfc-editor.org/info/rfc5015>.

[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, <http://www.rfc-editor.org/info/rfc6514>.

[RFC7441] Wijnands, IJ., Rosen, E., and U. Joorde, "Encoding Multipoint LDP (mLDP) Forwarding Equivalence Classes (FECs) in the NLRI of BGP MCAST-VPN Routes", RFC 7441, DOI 10.17487/RFC7441, January 2015, <http://www.rfc-editor.org/info/rfc7441>.

5.2. Informative References

[I-D.wijnands-bier-mld-lan-election]
Wijnands, I., Pfister, P., and Z. Zhang, "Generic Multicast Router Election on LAN's", draft-wijnands-bier-mld-lan-election-01 (work in progress), July 2016.

[RFC6826] Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M. Napierala, "Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013, <http://www.rfc-editor.org/info/rfc6826>.

[RFC7431] Karan, A., Filsfils, C., Wijnands, IJ., Ed., and B. Decraene, "Multicast-Only Fast Reroute", RFC 7431, DOI 10.17487/RFC7431, August 2015, <http://www.rfc-editor.org/info/rfc7431>.

[RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <http://www.rfc-editor.org/info/rfc7938>.

Authors' Addresses

Zhaohui Zhang
Juniper Networks

EMail: zzhang@juniper.net

Keyur Patel
Arrcus

EMail: keyur@arrcus.com

IJsbrand Wijnands
Cisco Systems

EMail: ice@cisco.com

Arkadiy Gulko
Thomson Reuters

EMail: arkadiy.gulko@thomsonreuters.com