L2VPN Working Group
Internet Draft
Intended status: Standards Track
Expires: January 2015

Dave Allan, Jeff Tantsura
Ericsson
Sam Aldrin
Huawei

July 2014

mLDP extensions for integrating EVPN and multicast
draft-allan-l2vpn-mldp-evpn-03

Abstract

This document describes how mLDP FECs can be encoded to support both service specific and shared multicast trees and describes the associated procedures for EVPN PEs. Thus, mLDP can implement multicast for EVPN.

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".

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 July 2014.

Copyright and License Notice

Table of Contents

1. Introduction

This document describes how mLDP FECs can be encoded to permit mLDP to implement multicast for EVPN. Such support can be applied to interconnecting 802.1ad, 802.1ah, 802.1aq, and 802.1Qbp based networks.

1.1. Authors

David Allan, Jeff Tantsura, Sam Aldrin

1.2. 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 [1].

2. Changes since last version

1) Added co-author.

3. Conventions used in this document

3.1. Terminology

BCB: Backbone Core Bridge
BEB: Backbone Edge Bridge
BU: Broadcast/Unknown
B-MAC: Backbone MAC Address
B-VID: Backbone VLAN ID
CE: Customer Edge
C-MAC: Customer/Client MAC Address
DF: Designated Forwarder
ESI: Ethernet segment identifier
EVPN: Ethernet VPN
FEC: Forwarding Equivalence Class
ISIS-SPB: IS-IS as extended for SPB
I-SID: Backbone Service Instance ID
mLDP: Multicast Label Distribution Protocol
MP2MP: Multipoint to Multipoint
MVPN: Multicast VPN
NLRI: Network layer reachability information
PBBN: Provider Backbone Bridged Network
BEB-PE: Co located BEB and PE
PE: provider edge
P2MP: Point to Multipoint
P2P: Point to Point

RD: Route Distinguisher
SPB: Shortest path bridging
SPBM: Shortest path bridging MAC mode
VID: VLAN ID
VLAN: Virtual LAN

4. Solution Overview

mLDP[6] permits arbitrary FEC encodings for the naming of multicast trees to be defined. This property is leveraged to permit both service specific trees and shared trees to be utilized to augment EVPN unicast connectivity with network based multicast and avoid the inefficiencies of edge replication.
The flooding of EVPN BGP NLRI and ISIS-SPB [7] provides each PE with sufficient information to self elect as a DF, have knowledge of peer DFs, and from that construct the identifiers for the required set of multicast trees to support the current service set, which can then be encoded as mLDP FECs, and used to originate label mapping and label withdraw messages.
Both p2mp and mp2mp trees are supported with different FEC encodings for each. Service specific tree FECs encode the VID or I-SID associated with the service instance in the subtending network. Shared tree FECs encode a sorted list of the IP addresses of the leaf DFs.

5. Elements of Procedure

A PE advertises whether or not it supports shared tree (actual mechanism is TBD). Support of both shared and service specific trees is mandatory. Whether a PE supports shared trees is a network design decision.

A PE is expected to maintain a list of current multicast memberships.

A PE, upon receipt of new information from BGP or ISIS-SPB:

1) Evaluates it"s DF roles (as described in [5]).

2) On the basis of the PE"s DF role, determines the set of services it needs to support.

3) Determines the set of peer DFs for each service.

4) On the basis of requisite tree types and ESI multicast registrations (p2mp or mp2mp/service specific or shared), determines the name of the multicast tree needed for the service.

For example an ESI may only have source interest in an ISIS-SPB I-SID in which case it would:

- require a p2mp tree to the set of DFs registering receive interest in the I-SID for p2mp trees

- require an upstream label mapping to the set of DFs registering receive interest in the I_SID for mp2mp trees

5) Upon completion of evaluating the set of services, de-duplicates the required tree membership list.

6) Compares the required list with the existing list, and originates the necessary label mapping and label withdraw transactions to the network state up to date.

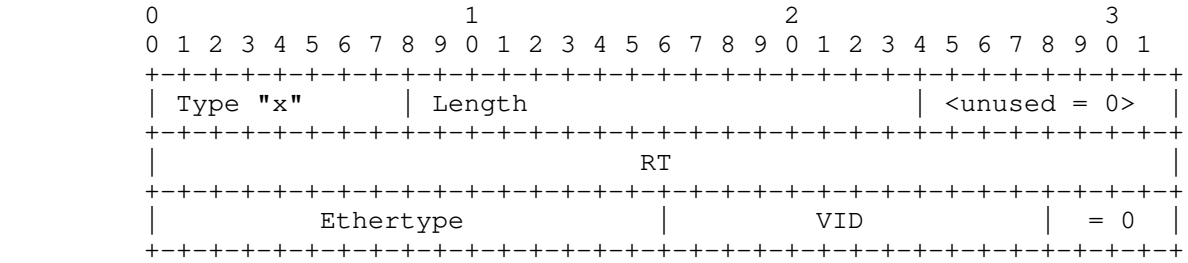
7) Configures the dataplane for the appropriate service to multicast tree bindings.

6. FEC Encoding

6.1. VLAN tagged FEC

VLAN tagged FEC uses the mLDP p2mp (0x06) type FEC and the mLDP mp2mp downstream (0x07) and upstream FECs (0x08)

The encoding of the opaque value is:

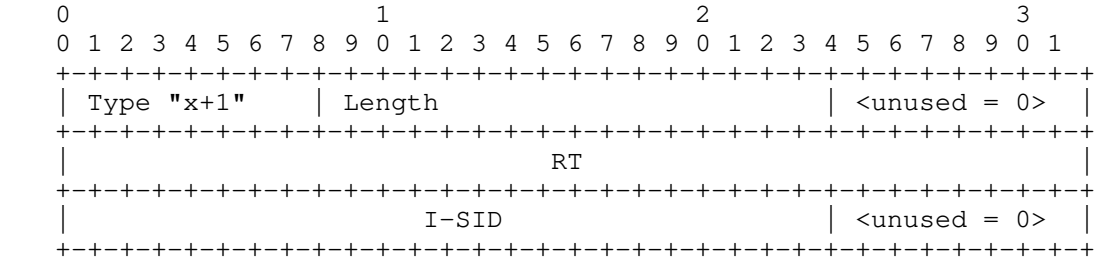


Where:

- RT is the route target for the EVPN instance
- Ethertype identifies the tag type (C 0x8100, S or B 0x88a8)
- VID is the VLAN ID tag value. If the VID=0, then this is the default MDT for the RT and how VLAN unaware RTs are encoded, else it permits MDTs to be defined for VLAN aware services.

6.2. I-SID tagged FEC

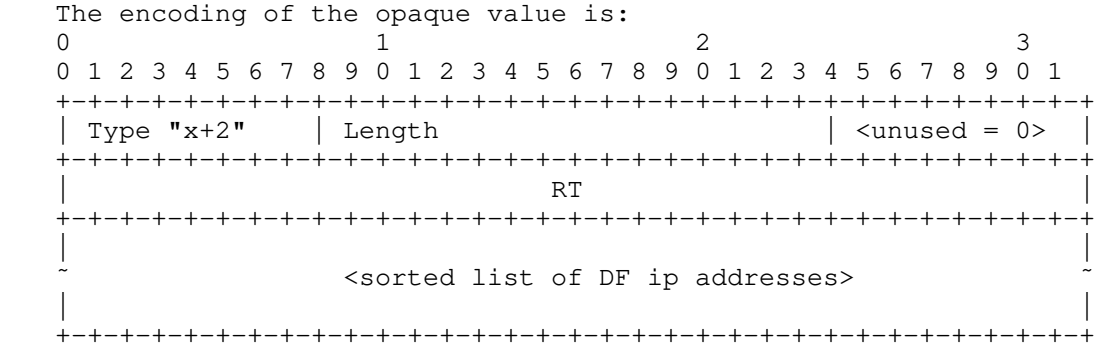
The encoding of the opaque value is:



Where:
- RT is the route target for the EVPN instance
- I-SID corresponds to the I-SID that will use the tree

6.3. Shared FEC

The encoding of the opaque value is:



Where:
- RT is the route target for the EVPN instance
- Sorted list of DF addresses identifies the set of leaves that have registered interest in one or more Ethernet services (either C/S or I tagged).

7. Acknowledgements

The authors would like to thank Panagiotis Saltsidis, Jakob Heitz, Don Fedyk and Janos Farkas for their detailed review of this draft.

8. Security Considerations

For a future version of this document.

9. IANA Considerations

For a future version of this document.

10. References

10.1. Normative References

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

[2] Fedyk et.al. "IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging", IETF RFC 6329, April 2012

[3] Rosen et.al., "BGP/MPLS IP Virtual Private Networks (VPNs)", IETF RFC 4364, February 2006

[4] Aggarwal et.al. "BGP MPLS Based Ethernet VPN", IETF work in progress, draft-ietf-l2vpn-evpn-01, July 2012

[5] Allan et.al. "802.1aq and 802.1Qbp Support over EVPN", IETF work in progress, draft-allan-l2vpn-spb-evpn-03, February 2013

[6] Wijnands et.al. "Label Distribution Protocol Extensions for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths". IETF RFC 6388, November 2011

10.2. Informative References

[7] IEEE 802.1aq "IEEE Standard for Local and Metropolitan Area Networks: Bridges and Virtual Bridged Local Area Networks - Amendment 9: Shortest Path Bridging", June 2012

[8] IEEE 802.1Qbp "Draft IEEE Standard for Local and Metropolitan Area Networks---Virtual Bridged Local Area Networks - Amendment: Equal Cost Multiple Paths (ECMP), 802.1Qbp", draft 1.3, February 2013

[9] Sajassi et.al. "PBB E-VPN", IETF work in progress, draft-ietf-l2vpn-pbb-evpn-03, June 2012

[10] IEEE 802.1Q-2011 "IEEE Standard for Local and metropolitan area networks--Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", August 2011

11. Authors' Addresses

Dave Allan (editor)
Ericsson
300 Holger Way
San Jose, CA 95134
USA
Email: david.i.allan@ericsson.com

Jeff Tantsura
Ericsson
300 Holger Way
San Jose, CA 95134
Email: jeff.tantsura@ericsson.com

Sam Aldrin
Huawei Technologies
2330 Central Expressway
Santa Clara, CA 95051
EMail: aldrin.ietf@gmail.com

INTERNET-DRAFT
Intended Status: Standard Track

Sami Boutros
Ali Sajassi
Samer Salam
Cisco Systems

John Drake
Juniper Networks

Jeff Tantsura
Ericsson

Dirk Steinberg
Steinberg Consulting

Thomas Beckhaus
Deutsche Telecom

J. Rabadan
Alcatel-Lucent

Expires: April 27, 2015

October 24, 2014

VPWS support in EVPN
draft-boutros-l2vpn-evpn-vpws-06.txt

Abstract

This document describes how EVPN can be used to support virtual private wire service (VPWS) in MPLS/IP networks. EVPN enables the following characteristics for VPWS: single-active as well as all-active multi-homing with flow-based load-balancing, eliminates the need for single-segment and multi-segment PW signaling, and provides fast protection using data-plane prefix independent convergence upon node or link failure.

Copyright and License Notice

Table of Contents

1 Introduction

This document describes how EVPN can be used to support virtual private wire service (VPWS) in MPLS/IP networks. The use of EVPN mechanisms for VPWS brings the benefits of EVPN to p2p services. These benefits include single-active redundancy as well as all-active redundancy with flow-based load-balancing. Furthermore, the use of EVPN for VPWS eliminates the need for signaling single-segment and multi-segment PWs for p2p Ethernet services.

[EVPN] has the ability to forward customer traffic to/from a given customer Attachment Circuit (AC), aka Ethernet Segment in EVPN terminology, without any MAC lookup. This capability is ideal in providing p2p services (aka VPWS services). [MEF] defines Ethernet Virtual Private Line (EVPL) service as p2p service between a pair of ACs (designated by VLANs) and Ethernet Private Line (EPL) service, in which all traffic flows are between a single pair of ESes. EVPL can be considered as a VPWS with only two ACs. In delivering an EVPL service, the traffic forwarding capability of EVPN based on the exchange of a pair of Ethernet AD routes is used; whereas, for more general VPWS, traffic forwarding capability of EVPN based on the exchange of a group of Ethernet AD routes (one Ethernet AD route per AC/segment) is used. In a VPWS service, the traffic from an originating Ethernet Segment can be forwarded only to a single destination Ethernet Segment; hence, no MAC lookup is needed and the MPLS label associated with the per-EVI Ethernet AD route can be used in forwarding user traffic to the destination AC.

Both services are supported by using the Ethernet A-D per EVI route which contains an Ethernet Segment Identifier, in which the customer ES is encoded, and an Ethernet Tag, in which the VPWS service instance identifier is encoded. I.e., for both EPL and EVPL services, a specific VPWS service instance is identified by a pair of Ethernet A-D per EVI routes which together identify the VPWS service instance endpoints and the VPWS service instance. In the control plane the VPWS service instance is identified using the VPWS service instance identifiers advertised by each PE and in the data plane the MPLS label advertised by one PE is used by the other PE to send traffic for that VPWS service instance. As with the Ethernet Tag in standard EVPN, the VPWS service instance identifier has uniqueness within an EVPN instance. Unlike standard EVPN where the Ethernet Tag MUST be carried in the MPLS encapsulated frames, VPWS does not use the Ethernet Tag value in the data plane. The MPLS label is enough to identify the VPWS service instance and provide egress tag translation at the disposition PE, if required. The Ethernet Segment identifier encoded in the Ethernet A-D per EVI route is not used to identify the service, however it can be used for flow-based load-balancing and mass withdraw functions.

As with standard EVPN, the Ethernet A-D per ES route is used for fast convergence upon link or node failure and the Ethernet Segment route is used for auto-discovery of the PEs attached to a given multi-homed CE and to synchronize state between them.

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 [RFC2119].

MAC: Media Access Control

MPLS: Multi Protocol Label Switching.

OAM: Operations, Administration and Maintenance.

PE: Provide Edge Node.

CE: Customer Edge device e.g., host or router or switch.

EVPL: Ethernet Virtual Private Line.

EPL: Ethernet Private Line.

VPWS: Virtual private wire service.

EVI: EVPN Instance.

Single-Active Mode: When a device or a network is multi-homed to two or more PEs and when only a single PE in such redundancy group can forward traffic to/from the multi-homed device or network for a given VLAN, then such multi-homing or redundancy is referred to as "Single-Active".

All-Active: When a device is multi-homed to two or more PEs and when all PEs in such redundancy group can forward traffic to/from the multi-homed device for a given VLAN, then such multi-homing or redundancy is referred to as "All-Active".

1.2 Requirements

1. EPL service access circuit maps to the whole Ethernet port.

2. EVPL service access circuits are VLANs on single or double tagged trunk ports. Each VLAN individually will be considered to be an endpoint for an EVPL service, without any direct dependency on any other VLANs on the trunk. Other VLANs on the same trunk could also be

used for EVPL services, but could also be associated with other services.

3. If multiple VLANs on the same trunk are associated with EVPL services, the respective remote endpoints of these EVPLs could be dispersed across any number of PEs, i.e. different VLANs may lead to different destinations.

4. The VLAN tag on the access trunk only has PE-local significance. The VLAN tag on the remote end could be different, and could also be double tagged when the other side is single tagged.

5. Also, multiple EVPL service VLANs on the same trunk could belong to the same EVPN instance (EVI), or they could belong to different EVIs. This should be purely an administrative choice of the network operator.

6. A given access trunk could have hundreds of EVPL services, and a given PE could have thousands of EVPLs configured. It must be possible to configure multiple EVPL services within the same EVI.

7. Local access circuits configured to belong to a given EVPN instance could also belong to different physical access trunks.

8. EPL-LAN and EVP-LAN are possible on the same system and also ESIs can be shared between EVPL and EVP-LANs.

2. BGP Extensions

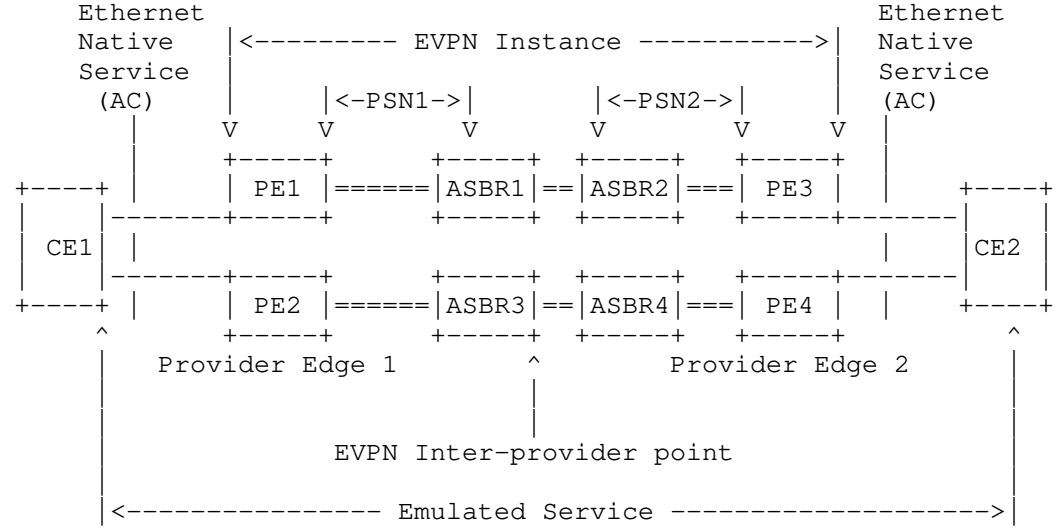
This document proposes the use of the Ethernet A-D per EVI route to signal VPWS services. The Ethernet Segment Identifier field is set to the customer ES and the Ethernet Tag ID 32-bit field is set to the 24-bit VPWS service instance identifier. For both EPL and EVPL services, for a given VPWS service instance the pair of PEs instantiating that VPWS service instance will each advertise an Ethernet A-D per EVI route with its VPWS service instance identifier and will each be configured with the other PE's VPWS service instance identifier. When each PE has received the other PE's Ethernet A-D per EVI route the VPWS service instance is instantiated. It should be noted that the same VPWS service instance identifier may be configured on both PEs.

The Route-Target (RT) extended community with which the Ethernet A-D per EVI route is tagged identifies the EVPN instance in which the VPWS service instance is configured. It is the operator's choice as to how many and which VPWS service instances are configured in a

given EVPN instance. However, a given EVPN instance MUST NOT be configured with both VPWS service instances and standard EVPN multi-point services.

3 Operation

The following figure shows an example of a P2P service deployed with EVPN.



iBGP sessions are established between PE1, PE2, ASBR1 and ASBR3, possibly via a BGP route-reflector. Similarly, iBGP sessions are established between PE3, PE4, ASBR2 and ASBR4. eBGP sessions are established among ASBR1, ASBR2, ASBR3, and ASBR4.

All PEs and ASBRs are enabled for the EVPN SAFI and exchange Ethernet A-D per EVI routes, one route per VPWS service instance. For inter-AS option B, the ASBRs re-advertise these routes with Next Hop attribute set to their IP addresses. The link between the CE and the PE is either a C-tagged or S-tagged interface, as described in [802.1Q], that can carry a single VLAN tag or two nested VLAN tags and it is configured as a trunk with multiple VLANs, one per VPWS service instance. It should be noted that the VLAN ID used by the customer at either end of a VPWS service instance to identify that service instance may be different and EVPN doesn't perform that translation between the two values. Rather, the MPLS label will identify the VPWS service instance and if translation is needed, it should be done by the Ethernet interface for each service.

For single-homed CE, in an advertised Ethernet A-D per EVI route the

ESI field is set to 0 and the Ethernet Tag field is set to the VPWS service instance identifier that identifies the EVPL or EPL service.

For a multi-homed CE, in an advertised Ethernet A-D per EVI route the ESI field is set to the CE's ESI and the Ethernet Tag field is set to the VPWS service instance identifier, which MUST have the same value on all PEs attached to that ES. This allows an ingress PE to perform flow-based load-balancing of traffic flows to all of the PEs attached to that ES. In all cases traffic follows the transport paths, which may be asymmetric.

The VPWS service instance identifier encoded in the Ethernet Tag field in an advertised Ethernet A-D per EVI route MUST either be unique across all ASs, or an ASBR needs to perform a translation when the Ethernet A-D per EVI route is re-advertised by the ASBR from one AS to the other AS.

Ethernet A-D per ES route can be used for mass withdraw to withdraw all Ethernet A-D per EVI routes associated with the multi-home site on a given PE.

4 EVPN Comparison to PW Signaling

In EVPN, service endpoint discovery and label signaling are done concurrently using BGP. Whereas, with VPWS based on [RFC4448], label signaling is done via LDP and service endpoint discovery is either through manual provisioning or through BGP.

In existing implementation of VPWS using pseudowires(PWs), redundancy is limited to single-active mode, while with EVPN implementation of VPWS both single-active and all-active redundancy modes can be supported.

In existing implementation with PWs, backup PWs are not used to carry traffic, while with EVPN, traffic can be load-balanced among different PEs multi-homed to a single CE.

Upon link or node failure, EVPN can trigger failover with the withdrawal of a single BGP route per EVPL service or multiple EVPL services, whereas with VPWS PW redundancy, the failover sequence requires exchange of two control plane messages: one message to deactivate the group of primary PWs and a second message to activate the group of backup PWs associated with the access link. Finally, EVPN may employ data plane local repair mechanisms not available in VPWS.

5 ESI Bandwidth

The ESI Bandwidth will be encoded using the Link Bandwidth Extended community defined in [draft-ietf-idr-link-bandwidth] and associated with the Ethernet AD route used to realize the EVPL services.

When a PE receives this attribute for a given EVPL it MUST request the required bandwidth from the PSN towards the other EVPL service destination PE originating the message. When resources are allocated from the PSN for a given EVPL service, then the PSN SHOULD account for the Bandwidth requested by this EVPL service.

In the case where PSN resources are not available, the PE receiving this attribute MUST re-send its local Ethernet AD routes for this EVPL service with the ESI Bandwidth = All FFs to declare that the "PSN Resources Unavailable".

The scope of the ESI Bandwidth is limited to only one Autonomous System.

6 Failure Scenarios

On a link or port failure between the CE and the PE for both single and multi-homed CEs, the PE must withdraw all the associated Ethernet AD routes for the VPWS service instances on the failed port or link.

6.1 Single-Homed CEs

Unlike [EVPN], EVPN-VPWS uses Ethernet AD route advertisements for single-homed Ethernet Segments. Therefore, upon a link/port failure of this single-homed Ethernet Segment, the PE MUST withdraw the associated Ethernet A-D routes.

6.1 Multi-Homed CEs

For a faster convergence in multi-homed scenarios with either Single-Active Redundancy or All-active redundancy, mass withdraw technique as per [EVPN] baseline is used. A PE previously advertising an Ethernet A-D per ES route, can withdraw this route signaling to the remote PEs to switch all the VPWS service instances associated with this multi-homed ES to the backup PE

7 VPWS with multiple sites

The VPWS among multiple sites (full mesh of P2P connections - one per pair of sites) that can be setup automatically without any explicit provisioning of P2P connections among the sites is outside the scope of this document.

8 Acknowledgements

The authors would like to acknowledge Wen Lin contributions to this document.

9 Security Considerations

This document does not introduce any additional security constraints.

10 IANA Considerations

TBD.

11 References

11.1 Normative References

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

11.2 Informative References

[RFC7209] A. Sajassi, R. Aggarwal et. al., "Requirements for Ethernet VPN".

[EVPN] A. Sajassi, R. Aggarwal et. al., "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-11.txt.

[PBB-EVPN] A. Sajassi et. al., "PBB-EVPN", draft-ietf-l2vpn-pbb-evpn-08.txt.

[draft-ietf-idr-link-bandwidth] P. Mohapatra, R. Fernando, "BGP Link Bandwidth Extended Community", draft-ietf-idr-link-bandwidth-06.txt

Authors' Addresses

Sami Boutros
Cisco
Email: sboutros@cisco.com

Ali Sajassi
Cisco
Email: sajassi@cisco.com

Samer Salam
Cisco

Email: ssalam@cisco.com

John Drake
Juniper Networks
Email: jdrake@juniper.net

Jeff Tantsura
Ericsson
Email: jeff.tantsura@ericsson.com

Dirk Steinberg
Steinberg Consulting
Email: dws@steinbergnet.net

Patrice Brissette
Cisco
Email: pbrisset@cisco.com

Thomas Beckhaus
Deutsche Telecom
Email:Thomas.Beckhaus@telekom.de>

Jorge Rabadan
Alcatel-Lucent
Email: jorge.rabadan@alcatel-lucent.com

Layer 3 IP VPN
Internet Draft
Intended status: Standard track
Updates: 6514, 6625
Expires: April 2015

Andrew Dolganow
Jayant Kotalwar
Alcatel-Lucent

October 16, 2014

Explicit tracking in MPLS/BGP IP VPNs
draft-dolganow-l3vpn-mvpn-expl-track-00.txt

Status of this Memo

Abstract

RFC 6513 and RFC 6514 define encoding and procedures for multicast MPLS/BGP IP Virtual Private Networks (VPNs). As defined in these RFCs, in some cases when BGP is used to exchange C-multicast routes, a need may exist to explicitly track PEs joining the same C-multicast-tree. The RFCs define encoding and procedures for explicit tracking using "PMSI Tunnel attribute" and "Leaf A-D route"; however, the procedures are missing details and clarity.

RFC 6625 defines encodings and procedures for wildcards in multicast MPLS/BGP IP VPNs. The RFC does not cover explicit tracking for wildcard multicast streams.

This documents updates RFC 6514 and 6625 with procedures required to achieve explicit PE tracking for (C-S, C-G) and wildcard multicast streams.

Table of Contents

1. Introduction

Section 5.3.2 of [MVPN] describes the method for explicit tracking of PEs that join the same C-tree. A mechanism using "PMSI Tunnel attribute" and "Leaf A-D routes" is described, while for detailed procedures a reader is referred to [MVPN-BGP].

[MVPN-BGP] defines encoding and procedures for, among others, explicit tracking in:

1. Section 5. This section defines how to encode "PMSI Tunnel attribute" to initiate explicit tracking on a PE: the attribute must have "Leaf Information required" (L) flag set to 1, Tunnel Type field set to "No tunnel information".

2. Section 4.4. This section defines encoding of "Leaf A-D Route" attribute that is used by PEs to respond to explicit tracking requests as encoded above. The section then points to procedures in other sections (especially section 12.3) of [MVPN-BGP] on how to generate and process "Leaf A-D route"

Following the above-procedures demonstrates that explicit tracking has not been fully incorporated into [MVPN-BGP] procedures. Therefore, a need exists to clarify and modify [MVPN-BGP] procedures for explicit tracking of PEs that join a given C-tree when BGP is used to exchange multicast routes. This document explicitly defines procedure clarification/modification required to achieve explicit PE tracking.

1.1. Terminology

This document uses terminology from [MVPN] and [MVPN-BGP] and [MVPN-WC].

2. Conventions used in this document

Wildcard streams term applies to each (C-*, C-G), (C-S, C-*) and (C-*, G-*) types of wildcard multicast streams as defined in [MVPN-WC] unless explicitly limited to only a subset of those wildcard types.

Multicast stream term applies to (C-S, C-G) or a wildcard stream as defined above.

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 Error! Reference source not found..

3. Explicit tracking request encoding

3.1. S-PMSI A-D route without tunnel information

This case applies when S-PMSI A-D route does not exist OR S-PMSI A-D route's Multicast Source and Multicast Group fields do not encode the stream to be tracked.

A PE originating explicit tracking for the multicast stream MUST use MCAST-VPN NLRI of an S-PMSI A-D route with PMSI tunnel attribute encoded as per section 5 of [MVPN-BGP] including setting "Leaf Information required" (L) flag to 1 and Tunnel Type field set to "No tunnel information".

3.2. S-PMSI A-D route with tunnel information

This case applies when S-PMSI A-D route's Multicast Source and Multicast Group fields encode the stream to be tracked.

A PE originating explicit tracking for multicast stream SHOULD set the Leaf Information Required flag in the PMSI Tunnel attribute of the MCAST-VPN NLRI of the S-PMSI A-D route to 1 and keep all other fields unchanged. This ensures that this PE originates only a single S-PMSI A-D route for this (C-S, C-G) or wildcard stream.

4. Intra-AS explicit tracking procedures

The following sections define procedures on PEs to support explicit tracking for Intra-AS.

4.1. Procedures on Sender PE

4.1.1. S-PMSI A-D route without tunnel information

To initiate explicit tracking for a (C-S, C-G) stream, a PE must originate S-PMSI A-D route as defined by procedures in section 12.1 of [MVPN-BGP] with the following modification:

1. PMSI tunnel attribute MUST be encoded as per section 4.1 above (i.e. the attribute does not contain P-multicast tree tunnel information)

To initiate explicit tracking for a wildcard stream, a PE must originate an S-PMSI A-D route as defined by procedures in section 4 of [MVPN-WC] with the following modification:

1. PMSI tunnel attribute MUST be encoded as per section 4.1 above (i.e. the attribute does not contain P-multicast tree identity)

2. PE MUST exclude the above-generated S-PMSI A-D route when finding a match for Data transmission as specified in section 3.1 of [MVPN-WC]

Upon receiving a Leaf A-D route in response to the explicit tracking request for a multicast stream, the PE performs regular procedures defined in section 9.2.3.5 of [MVPN-BGP]. Specific use of tracking information on the PE is out of scope for this specification.

To terminate explicit tracking for multicast stream, a PE MUST withdraw the above-originated S-PMSI A-D route.

If the sender PE decides to bound explicitly tracked stream to S-PMSI P-tunnel, the PE MUST execute procedures defined in section 12.1 of [MVPN-BGP] for (C-S, C-G) and, if applicable, procedures of section 4 of [MVPN-WC]. These procedures, among others, will re-originate S-PMSI A-D route with updated PMSI Tunnel attribute (including encoding of S-PMSI P-multicast tree tunnel information). If explicit tracking for the multicast stream is to continue, the resulting S-PMSI A-D route type will be encoded as per section 3.2 above.

4.1.2. S-PMSI A-D route with tunnel information

The procedures defined in this section apply only if PE has originated S-PMSI A-D route with PMSI Tunnel attribute encoding the multicast stream to be tracked with Leaf Information Required flag set to 0.

To initiate explicit tracking for the multicast stream, a PE SHOULD update the PMSI tunnel attribute with Leaf Information Required flag to 1 and re-originate the S-PMSI A-D route

To stop explicit tracking for the multicast stream, a PE SHOULD re-originate S-PMSI A-D route with all information unchanged but Leaf Information Required flag set to 0.

To continue explicit tracking for a multicast stream which is being switched from an S-PMSI tunnel to an I-PMSI tunnel, the PE should first execute procedures to initiate explicit tracking of a multicast stream as defined in section 4.1.1 above to ensure tracking information remains current during the switch.

4.2. Procedures on PE receiving S-PMSI route with explicit tracking "PMSI Tunnel attribute"

We say a PE is to receive traffic for the explicitly tracked wildcard stream, if at least one (C-S, C-G) C-flow matches the S-PMSI A-D route encoding the explicitly tracked stream using procedures of section 3.2 of [MVPN-WC].

PE receiving S-MPSI A-D route with tracking request, performs procedures defined in section 12.3 of [MVPN-NG] either as result of receiving (C-S, C-G) tracking request or as consequence of procedures of section 4 of [MVPN-WC] for wildcard stream tracking request with the following modifications:

1. The procedures requiring set-up of forwarding path to receive traffic from the tunnel advertised by the S-PMSI route do not apply if the PMSI Tunnel attribute is encoded as per section 3.1 of this document. The PE continues to receive traffic on the current P-tunnel.

2. If a PE determines that it no longer is to receive traffic for an explicitly tracked multicast stream from the PE that originated explicit tracking for that streams, the PE MUST withdraw the Leaf A-D route originated in response to explicit tracking using standard procedures defined in [MVPN-BGP].

3. If a PE determines that it is to receive traffic for an explicitly tracked multicast stream from the PE that originated explicit tracking for that streams, the PE MUST re-originate a Leaf A-D route as defined in [MVPN-BGP].

5. Inter-AS procedures

Left for future study

6. Security Considerations

No security considerations beyond those already covered by [MVPN], [MVPN-BGP] and [MVPN-WC] are introduced by this document.

7. IANA Considerations

8. Conclusions

<Add any conclusions>

9. References

9.1. Normative References

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

[MVPN-BGP]"BGP encodings and Procedures for Multicast in MPLS/BGP IP VPNs", R. Aggarwa, E. Rosen, T. Morin, and Y. Rekhter, RFC 6514, February 2012

[MVPN-WC] "Wildcards in Multicast VPN Auto-Discovery Routes", E. Rosen, Y. Rekhter, W. Hendrickx, R. Qiu, RFC 6625, May 2012

[PIM-SM] "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", B. Fenner, M. Handley, H. Holbrook,and I. Kouvelas, RFC 4601, 2006

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

9.2. Informative References

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

Authors' Addresses

Andrew Dolganow
Alcatel-Lucent
600 March Rd.
Ottawa, ON, Canada
Email: andrew.dolganow@alcatel-lucent.com

Jayant Kotalwar
Alcatel-Lucent
701 East Middlefield Rd
Mountain View, CA 94043, USA
Email: jayant.kotalwar@alcatel-lucent.com

BESS Working Group W. Hao
L. Yong
S. Hares
Internet Draft Huawei
Osama Zia
Microsoft
Muhammad Durrani
Cisco
Intended status: Standard Track September 10, 2015
Expires: March 10, 2016

Inter-AS Option C between NVO3 and BGP/MPLS IP VPN network
draft-hao-bess-inter-nvo3-vpn-optionc-03.txt

Abstract

This draft describes inter-as option-C solution between NVO3 network and MPLS/IP VPN network. Transport layer stitching solution should be provided. Also to ensure VPNv4 route exchange correctly between local NVE and remote PE, VNID space should be partitioned, only the VNIDs of lower 1 Million can be used for interconnection with outer MPLS VPN network using option-C solution, the rest 15 Million VNIDs can only be used for intra DC.

Table of Contents

1. Introduction

In cloud computing era, multi-tenancy has become a core requirement for data centers. Since Network Virtualization Overlays (NVO3) can satisfy multi-tenancy key requirements, this technology is being deployed in an increasing number of cloud data center network. NVO3 focuses on the construction of overlay networks that operate over an IP (L3) underlay transport network. It can provide layer 2 bridging

and layer 3 IP service for each tenant. VXLAN [RFC7348] and NVGRE [NVGRE] are two typical NVO3 technologies. In NVO3 network, 24-bit VNID (or VSID) is used to identify different virtual networks, theoretically 16M virtual networks can be supported in a data center. MPLS Over GRE and MPLS In UDP [RFC7510] are another two technologies to construct the overlay network, 20-bit MPLS Label is used as virtual networks identification. NVO3 overlay network can be controlled through centralized NVE-NVA architecture or through distributed BGP VPN protocol.

NVO3 has good scaling properties from relatively small networks to networks with several million tenant systems (TSs) and hundreds of thousands of virtual networks within a single administrative domain. In a data center network, each tenant may include one or more layer 2 virtual network. In normal case, each tenant corresponds to one routing domain (RD), each layer 2 virtual network corresponds to one or more subnets.

To provide cloud service to external data center customers, data center networks should be connected to the WAN network. BGP MPLS/IP VPN are widely deployed technologies on WAN networks. Normally internal data center and external MPLS/IP VPN network are different Autonomous System (AS).

In multiple NVO3 data center inter-connecting scenario, the traffic across data center normally are carried over BGP MPLS/IP VPN network. This also requires an applicable inter-as solution between NVO3 network and external MPLS/IP network which can meet scale demands on existing and future NVO3 data center.

Similar to the Inter-as connection method defined in RFC4364, there are three different ways of handling this case, they are option-A, option-B and option-C respectively in order of increasing scalability.

Option-A is a back-to-back VRFs solution. Using option-A, EBGP session per VPN is created on peering ASBRs. In the data-plane, VLANs are used for tenant traffic separation. It has the lowest scalability among the three solutions. Compared to option-A solution, option-B solution has more scalability. But using option-B, ASBRs need to maintain and distribute all VPN prefixes. In the data plane, ASBRs need to perform MPLS VPN Label switching. Because MPLS VPN Label switching table space on ASBRs is limited, it still has scalability limitation for large VPN network. Option-C solution is a most scalable option through separating VPNv4 and PE prefixes exchange, the ASBRs don’t need to maintain and distribute the

customers VPN prefixes. The ASBR is only used to exchange the service provider(SP) internal IP.

This draft is to propose inter-as option-C solution between NVO3 network and external BGP MPLS/IP VPN network. Compared to the traditional option-C solution defined in [RFC4364], it is for heterogeneous network interconnection, the control plane and data plane procedures in NVO3 network should be newly specified.

2. Conventions used in this document

Network Virtualization Edge (NVE) - An NVE is the network entity that sits at the edge of an underlay network and implements network virtualization functions.

Tenant System - A physical or virtual system that can play the role of a host, or a forwarding element such as a router, switch, firewall, etc. It belongs to a single tenant and connects to one or more VNs of that tenant.

VNID - Virtual Network Identifier (for VxLAN)

VSID - Virtual Subnet Identifier (for NVGRE)

RD - Route Distinguisher. RDs are used to maintain uniqueness among identical routes in different VRFs, The route distinguisher is an 8-octet field prefixed to the customer's IP address. The resulting 12-octet field is a unique "VPN-IPv4" address.

RT - Route targets. It is used to control the import and export of routes between different VRFs.

3. Reference model

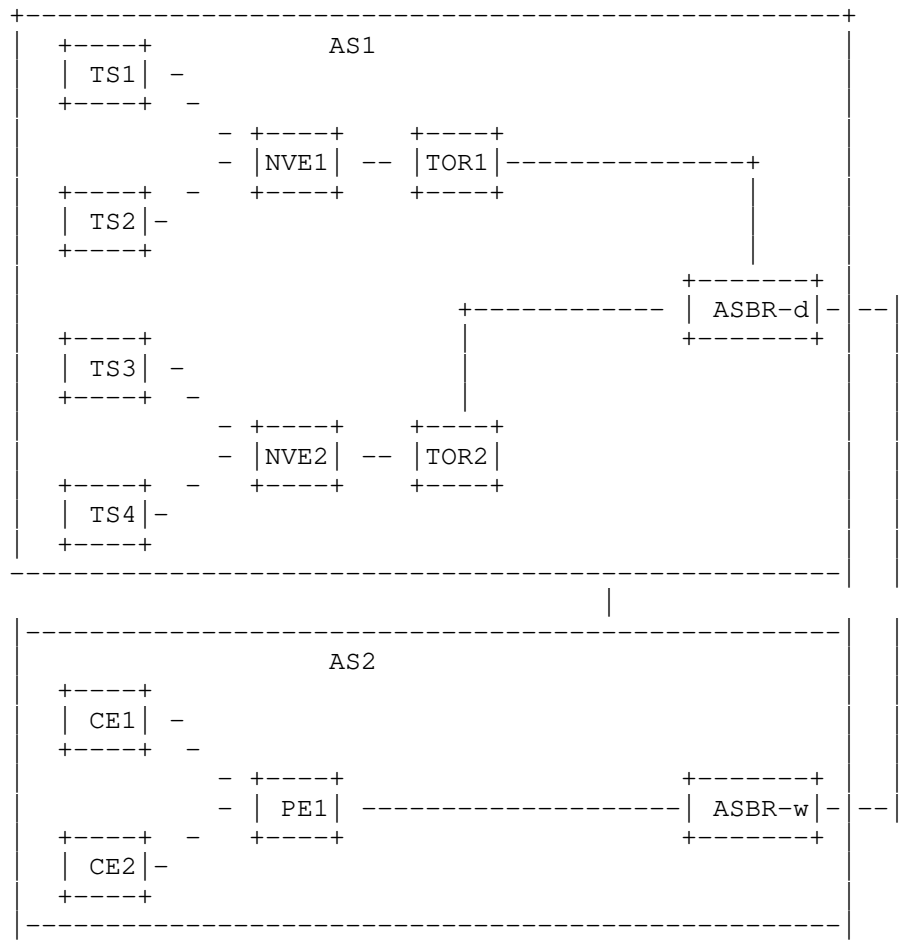


Figure 1 Reference model

Figure 1 shows an arbitrary Multi-AS VPN interconnectivity scenario between NVO3 network and BGP MPLS/IP VPN network. NVE1, NVE2, and ASBR-d forms NVO3 overlay network in internal DC. TS1 and TS2 connect to NVE1, TS3 and TS4 connect to NVE2. PE1 and ASBR-w forms MPLS IP/VPN network in external DC. CE1 and CE2 connect to PE1. The NVO3 network is in AS 1, the MPLS/IP VPN network is in AS 2.

There are two tenants in NVO3 network, TSs in tenant 1 can freely communicate with CEs in VPN-Red, TSs in tenant 2 can freely communicate with CEs in VPN-Green. TS1 and TS3 belong to tenant 1, TS2 and TS4 belong to tenant 2. CE1 belongs to VPN-Red, CE2 belongs

to VPN-Green. VNID 10 and VNID 20 are used to identify tenant1 and tenant2 respectively. PE1 assigned MPLS VPN Label 1000 and 2000 for the routes from CE1 and CE2 respectively.

4. Traditional Option-C [RFC4364] Recap

In traditional Option-C defined in [RFC4364], an MP-EBGP session between the end PEs in source and destination ASs is used for the redistribution of VPN-IPv4 routes. Labeled IPv4 routes are redistributed by EBGP between neighboring autonomous systems , inter-AS Option-C uses BGP as the label distribution protocol. Through this solution, VPN connectivity is maintained while keeping VPN-IPv4 routes out of the ASBRs, an ASBR only need maintain labeled IPv4/32 routes to the PE routers within its AS. If the /32 routes for the PE routers are NOT made known to the P routers(other than the ASBRs), then a packet's ingress PE need to put a three-label stack on it. The bottom label is assigned by the egress PE, corresponding to the packet's destination address in a particular VRF. The middle label is assigned by the ASBR, corresponding to the /32 route to the egress PE. The top label is assigned by the ingress PE's IGP Next Hop, corresponding to the /32 route to the ASBR.

5. Inter-As Option-C Solution

Each NVE operates as default layer 3 gateway to connect locally attached TS(s). VRFs are created on each NVE to isolate IP data plane forwarding table between various attached tenants. The VNID(or VSID and MPLS Label) is used as Tenant identification .

Similar to traditional Option-C defined in [RFC4364], an end to end tunnel path from NVE to PE as transport layer should be established through EBGPs (ASBR-ASBR) and two IBGPs in local DCs (between ASBR-PE and ASBR-NVE), and MP-BGP will be established over the tunnel so that VPN-IPv4 routes can be exchanged between the NVE and PE without AS awareness. Unlike traditional Option-C BGP label switched path, the tunnel path has two segments, one segment is the NVO3 tunnel from NVE to ASBR-d in NVO3 network, another segment is traditional BGP LSP from ASBR-d to PE in WAN network, the two segments should be stitched together at ASBR-d as per recommended implementation. The behavior on ASBR-w and PEs in MPLS VPN network has no implementation differences compared to the behavior of ASBR and PEs in traditional RFC4364 based MPLS VPN Option-C network.

5.1. EBGP process for transport layer stitching

This section will describe the EBGP procedures for the transport layer forwarding path stitching.

In WAN to DC direction, when ASBR-d receives labeled IPv4/32 routes from ASBR-w, one of the several allocation methods can be used for tunnel stitching among them few are IP, UDP port and GRE key allocation method . The method chosen by operator depends on the data center network type and the network scale. For the UDP based network of VXLAN [RFC7348] and MPLS In UDP [RFC7510], either IP allocation method or UDP port allocation method can be used. UDP port allocation should be within UDP ephemeral port range and one UDP port maps to a label. For NVGRE network, only IP allocation method can be used. For MPLS Over GRE network, either IP allocation method or GRE key allocation method can be used.

In DC to WAN direction, the transport layer stitching solution is same for all kinds of NVO3 network. In this solution, ASBR-d announces labeled IPv4/32 routes to ASBR-w for each NVE where unique MPLS Label is allocated. . The allocated MPLS Label and NVE IP address mapping forms incoming forwarding table which is used to stitch BGP LSP and NVO3 tunnel for inbound traffic forwarding, i.e., from external DC to internal DC.

5.1.1. UDP based overlay network

Both VXLAN and MPLS In UDP are UDP based encapsulations. For the outbound traffic from NVE to ASBR-d, there are two options at ASBR-d, i.e., the ASBR-d only accepts the traffic with standard destination UDP port(4789 for VXLAN [RFC7348], 6635 for MPLS In UDP [RFC7510]) or non-standardized destination UDP port in outer UDP header encapsulation.

In WAN to DC direction, if standard destination UDP port solution is used, when ASBR-d receives labeled IPv4/32 routes from ASBR-w, IP address allocation method should be used. The ASBR allocates an IP address per MPLS Label specified for a particular route defined in [RFC3107] to identify each remote PE, and then advertises the IPv4/32 route(indicating remote PE reachability) to all local NVEs with the VXLAN or MPLS In UDP tunnel attribute. [TUNNELENCAP] defines the relevant TLVs and sub-TLVs for the Tunnel Encapsulation Attribute. The local NVEs will encapsulate transport layer header using the Tunnel Encapsulation Attribute for the outbound traffic from internal DC to external DC, the ASBR-d generated IP is the

destination IP in NVO3 tunnel outer header, the UDP port is the standard well-known port for VXLAN and MPLS In UDP. The IP pool should be configured beforehand on ASBR-d. The new allocated IP and MPLS Label mapping forms outgoing forwarding table on ASBR-d which is used to stitch NVO3 tunnel and BGP LSP for outbound traffic forwarding . If non-standard destination UDP port is used, ASBR-d can allocate the combination of IP and UDP port(or only UDP port) per MPLS Label to identify each remote PE, and then advertises the IPv4/32 route received from ASBR-w to all local NVEs with the Tunnel Encapsulation Attribute. For each NVE, the destination IP and the destination port in NVO3 tunnel outer header is the new allocated IP and the new allocated port respectively. The new allocated IP and UDP port combination (or only UDP port) and MPLS Label mapping forms outgoing forwarding table on ASBR-d. This method is called UDP allocation method, the allocated UDP port range should be configured beforehand on ASBR-d.

In summary, IP allocation method has more IP address consumption than the UDP allocation method. If there is large number of remote PEs in WAN network, the UDP allocation method is suggested to be used to enhance network scalability.

5.1.2. GRE based overlay network

Both NVGRE and MPLS Over GRE are GRE based encapsulations. The GRE key field can be used to convey application-specific key value. In NVGRE, the key field has been used to convey 24-bit Virtual Subnet Identifier (VSID) as tenant identification, so for NVGRE, the GRE key field can’t be used for the stitching purpose and only IP allocation method can be used. In MPLS Over GRE, the GRE key field has not been used explicitly by an application and can be used for the transport layer stitching at ASBR-d, i.e., GRE key allocation method can be used to conserve IP address space.

In WAN to DC direction, for MPLS Over GRE, when ASBR-d receives labeled IPv4/32 routes from ASBR-w, the ASBR can allocate a GRE key per MPLS Label to identify each remote PE, and then advertises the IPv4/32 route to all local NVEs with the Tunnel Encapsulation Attribute. The new allocated GRE key and MPLS Label correspondence forms outgoing forwarding table on ASBR-d. This method is called GRE key allocation method.

If ASBR-d needs to change IP address, UDP port or GRE key for a particular /32 route, it should advertising a new route with the

same NLRI and a new Tunnel Encapsulation Attribute to refresh all NVEs's local information.

5.2. VPN routes exchange

Each NVE and remote PE should establish MP-EBGP session for the announcement of VPN-IPv4 routes through RFC4364. Route distinguishers (RD) and RT are specified for each VRF on each NVE and PE.

Each NVE advertises all local VPN route to remote PEs using tenant identification VNID (or VSID and MPLS Label) as MPLS VPN Label. These remote PEs deal with the NVE as regular PE, they match RT and populates these VPN route to local VRF. For the traffic from remote CE to local TS, ingress PE uses the VNID as bottom label in MPLS encapsulation. Because VNID field is 24 bits, to ensure these NVEs and PEs interworking, VNID length should not be beyond 20 bits, i.e., VNID value must not be larger than 1 Million. In proposed implementation NVO3 network the VNID space should be partitioned, only the VNIDs of lower 1 Million can be used for interconnection with outer MPLS VPN network, the rest 15 Million VNIDs can only be used for intra DC.

Each MPLS VPN PE also advertises all local VPN route to peer NVEs, these NVEs match RT and populates these VPN route to local VRF. For the traffic from local TS to remote CE, because ingress NVE doesn't support MPLS encapsulation, it encoded the MPLS VPN Label advertised from remote PE as VNID in NVO3 encapsulation.

5.3. Data forwarding process

When VXLAN network and UDP port allocation method are used in data center, the procedures of data forwarding between TS1 and CE1 in figure 1 will be described step by step as follows.

5.3.1. Data flow from TS1 to CE1

1. TS1 sends a packet to NVE1, destination IP is CE1's IP.

2. NVE1 acquires local VRF relying on packet input interface, then looks up the VRF's routing table corresponding to tenant 1, performs NVO3 encapsulation, and sends the encapsulated packet out to ASBR-d. The MPLS VPN Label associated with the packet's destination address is encoded in VNID field. VXLAN tunnel destination IP and destination UDP port are the IP address and UDP port allocated on ASBR-d associated with the /32 routes for the remote PE routers that the remote CE attached to.

3. ASBR-d decapsulates the VXLAN received packet and performs MPLS encapsulation. Two Labels should be pushed in the MPLS encapsulation, BGP LSP Label as top Label and MPLS VPN Label as bottom Label. BGP LSP Label is acquired by looking up outgoing stitching table, MPLS VPN Label is copied from VNID.

4. ASBR-w swaps BGP MPLS Label, and push IGP Label and sends the packet out to PE1. MPLS VPN Label remains unchanged.

5. PE1 pops all MPLS Label, finds local VRF relying on bottom MPLS VPN Label, performs looks up in local VRF IP forwarding table , and then sends the packet out to CE1.

5.3.2. Data flow from CE1 to TS1

1. CE1 sends a packet to PE1, destination IP is TS1's IP.

2. PE1 acquires local VRF interface relying on packet input interface where CE1 egress out the packet, then launches a lookup in VRF's routing table. It pushes three-label stack on the outgoing packet. The bottom label is the tenant VNID corresponds to TS1, the VNID is 10. The middle label is assigned by the ASBR-w, associating with the /32 route for the egress NVE1. The top label is assigned by the ingress PE's IGP Next Hop, corresponding to the /32 route to ASBR-w.

3. ASBR-w pops top IGP Label, swaps middle BGP Label, and then sends the packet out to ASBR-d.

4. ASBR-d decapsulates MPL packet, performs VXLAN encapsulation and then sends the packet to egress NVE1. The egress NVE's IP address is acquired by performing a looking up in the stitching table, VNID is copied from the bottom MPLS VPN Label.

5. NVE1 decapsulates incoming NVO3 encapsulated packet, looks up local VRF interface based on VNID, then performs a look up in the routing table and forwards the packet out to destination TS1.

6. NVE-NVA architecture

In this architecture, the NVE control plane and forwarding functionality are decoupled. All NVEs in NVO3 network don't need to support BGP protocol; these NVEs have only data plane functionality and are controlled by centralized NVA using openflow, ovsdb, i2rs, etc. The NVA runs BGP with ASBR-d to exchange plain IP route to

IPv4/32 of each WAN PE associated with the BGP tunnel encapsulation attribute. The NVA also runs MP-BGP protocol [RFC4364] with peer PE for all the NVEs to exchange VPNv4 route, VNID is used as MPLS VPN Label. ASBR-d can choose IP allocation, UDP allocation or GRE key allocation method for the transport stitching.

NVA maintains all tenant information, and originates BGP routes with the appropriate RD and RT. The NVA tenant information includes VNID(or VSID and MPLS Label) to identify each tenant and the corresponding RD and RT. This information can be statically configured by operators or dynamically allocated. This information also includes all TS's MAC/IP address and its attached NVE information.

6.1. EBGP process for transport layer stitching

DC to WAN direction:

1. ASBR-d allocates BGP MPLS Label per NVE.

2. ASBR-d advertises BGP Label routing information to peer ASBR-w. ASBR-d generates incoming stitching table <new allocated BGP MPLS Label, NVE IP>.

WAN to DC direction:

1. ASBR-d receives BGP Label routing information from peer ASBR-w.

2. ASBR-d allocates NVO3 Tunnel IP, UDP port or GRE key for each MPLS Label received from ASBR-w, the ASBR-d announces the IPv4/32 Route to NVA with the Tunnel Encapsulation Attribute.

3. ASBR-d generates outgoing stitching table<new allocated Tunnel IP(or UDP port and GRE key), received MPLS Label>.

6.2. VPN route exchange

NVA advertises all internal data center tenant routing information to remote PEs using RFC 4364, which includes RD, RT, IP prefix, and MPLS VPN Label, the tenant identification of VNID is used as MPLS VPN Label.

Each remote MPLS VPN PE also advertises local VPN routes to NVA. NVA acquires NVO3 Tunnel IP(or UDP port and GRE key) allocated by

ASBR-d corresponding to the PE, matches RT attribute and populates the VPN routes to local VRF.

Then the NVA downloads corresponding VPN forwarding table including <destination IP prefix/Mask, NVO3 Tunnel IP(or UDP port and GRE key), VNID>to each NVE.

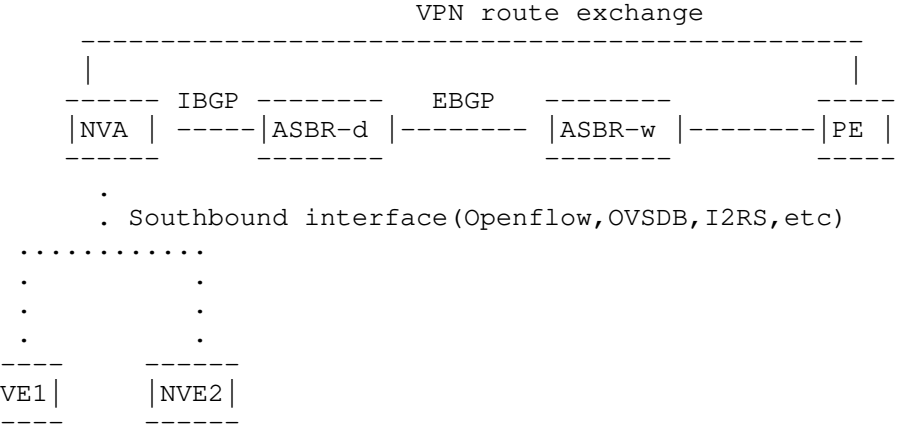


Figure 2 NVE-NVA Architecture

7. Security Considerations

Internal IP (Loopback IP for PE/NVE) addresses a network is advertised and visible in another network, which is a security risk. Most operators wants to prevent any external visibility and access into their internal devices IP. option C is suggested to be deployed within a single SP or enterprise with both MPLS and NVO3 networks.

8. IANA Considerations

NA.

9. References

9.1. Normative References

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

[2] [RFC4364] E. Rosen, Y. Rekhter, " BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[3] [RFC3107] Y. Rekhter,E. Rosen, ''Carrying Label Information in BGP-4'', RFC 3107, May 2001

9.2. Informative References

[1] [NVA] D.Black, etc, "An Architecture for Overlay Networks (NVO3)", draft-ietf-nvo3-arch-01, February 14, 2014

[2] [RFC7047] B. Pfaff, B. Davie,''The Open vSwitch Database Management Protocol'', RFC 7047, December 2013

[3] [OpenFlow1.3]OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04). June 25, 2012. (https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf)

[4] [RFC7348] M. Mahalingam, etc, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC7348, August 2014.

[5] [NVGRE] P. Garg, etc, "NVGRE: Network Virtualization using Generic Routing Encapsulation", draft-sridharan-virtualization-nvgre-08, April 13, 2015.

[6] [TUNNELENCAP] E. Rosen, etc, "Using the BGP Tunnel Encapsulation Attribute without the BGP Encapsulation SAFI", draft-rosen-idr-tunnel-encaps-00, June, 2015.

10. Acknowledgments

Authors like to thank Thomas Morin, Shunwan Zhuang, Haibo Wang, Jie Dong for their valuable inputs.

Authors' Addresses

Weiguo Hao
Huawei Technologies
101 Software Avenue,
Nanjing 210012
China
Phone: +86-25-56623144
Email: haoweiguo@huawei.com

Lucy Yong
Huawei Technologies
Phone: +1-918-808-1918
Email: lucy.yong@huawei.com

Susan Hares
Huawei Technologies
Phone: +1-734-604-0323
Email: shares@ndzh.com.

Osama Zia
Microsoft
Email: osamaz@microsoft.com

Muhammad Durrani
Cisco
Phone: +1-408-527-6921
Email: mdurrani@cisco.com

Network Working Group
Internet-Draft
Updates: 4761 (if approved)
Intended status: Standards Track
Expires: January 26, 2020

B. Kothari
Augtera Networks
K. Kompella
Juniper Networks
W. Henderickx
Nokia
F. Balus
Cisco
J. Uttaro
AT&T

July 25, 2019

BGP based Multi-homing in Virtual Private LAN Service
draft-ietf-bess-vpls-multihoming-05.txt

Abstract

Virtual Private LAN Service (VPLS) is a Layer 2 Virtual Private Network (VPN) that gives its customers the appearance that their sites are connected via a Local Area Network (LAN). It is often required for the Service Provider (SP) to give the customer redundant connectivity to some sites, often called "multi-homing". This memo shows how BGP-based multi-homing can be offered in the context of LDP and BGP VPLS solutions. This document updates RFC 4761 by defining new flags in the Control Flags field of the Layer2 Info Extended Community.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 26, 2020.

Table of Contents

1. Introduction

Virtual Private LAN Service (VPLS) is a Layer 2 Virtual Private Network (VPN) that gives its customers the appearance that their sites are connected via a Local Area Network (LAN). It is often required for a Service Provider (SP) to give the customer redundant connectivity to one or more sites, often called "multi-homing". [RFC4761] explains how VPLS can be offered using BGP for auto-discovery and signaling; section 3.5 of that document describes how multi-homing can be achieved in this context. [RFC6074] explains how VPLS can be offered using BGP for auto-discovery (BGP-AD) and [RFC4762] explains how VPLS can be offered using LDP for signaling. This document provides a BGP-based multi-homing solution applicable to both BGP and LDP VPLS technologies. Note that BGP MH can be used for LDP VPLS without the use of the BGP-AD solution.

Section 2 lays out some of the scenarios for multi-homing, other ways that this can be achieved, and some of the expectations of BGP-based multi-homing. Section 3 defines the components of BGP-based multi-homing, and the procedures required to achieve this.

1.1. General Terminology

Some general terminology is defined here; most is from [RFC4761], [RFC4762] or [RFC4364]. Terminology specific to this memo is introduced as needed in later sections.

A "Customer Edge" (CE) device, typically located on customer premises, connects to a "Provider Edge" (PE) device, which is owned and operated by the SP. A "Provider" (P) device is also owned and operated by the SP, but has no direct customer connections. A "VPLS Edge" (VE) device is a PE that offers VPLS services.

A VPLS domain represents a bridging domain per customer. A Route Target community as described in [RFC4360] is typically used to identify all the PE routers participating in a particular VPLS domain. A VPLS site is a grouping of ports on a PE that belong to the same VPLS domain. The terms "VPLS instance" and "VPLS domain" are used interchangeably in this document.

A VPLS site is a grouping of ports on a PE that belong to the same VPLS domain. The terms "VPLS instance" and "VPLS domain" are used interchangeably in this document.

If the CE devices that connect to a VPLS site's ports have connectivity to any other PE device then the VPLS site is called a multi-homed VPLS site. Otherwise, it is called a single-homed VPLS site. The ports are partitioned between VPLS sites such that each port is in no more than one VPLS site. The terms "VPLS site" and "CE site" are used interchangeably in this document.

A BGP VPLS NLRI for the base VPLS instance that has non-zero VE block offset, VE block size and label base is called as VE NLRI in this document. Each VPLS instance is uniquely identified by a VE-ID. VE-ID is carried in the BGP VPLS NLRI as specified in section 3.2.2 in [RFC4761].

A VPLS NLRI with value zero for the VE block offset, VE block size and label base is called as CE NLRI in this document.
Section Section 3.1 defines CE NLRI and provides more detail.

A Multi-homed (MH) site is uniquely identified by a CE-ID. Sites are referred to as local or remote depending on whether they are configured on the PE router in context or on one of the remote PE routers (network peers). A single-homed site can also be assigned a CE-ID, but it is not mandatory to configure a CE-ID for single-homed sites. Section Section 3.1 provides detail on CE-ID.

1.2. Conventions

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].

2. Background

This section describes various scenarios where multi-homing may be required, and the implications thereof. It also describes some of the singular properties of VPLS multi-homing, and what that means from both an operational point of view and an implementation point of view. There are other approaches for providing multi-homing such as Spanning Tree Protocol, and this document specifies use of BGP for multi-homing. Comprehensive comparison among the approaches is outside the scope of this document.

2.1. Scenarios

In Figure 1, CE1 is a VPLS CE that is dual-homed to both PE1 and PE2 for redundant connectivity.

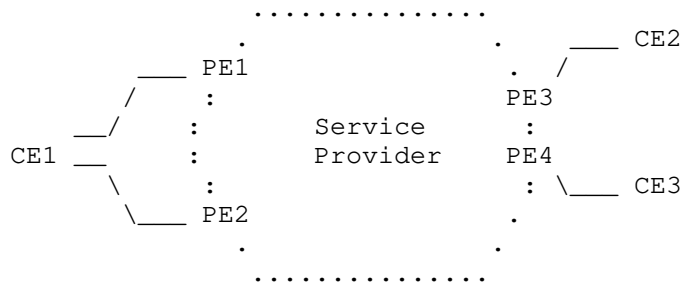


Figure 1: Scenario 1

In Figure 2, CE1 is a VPLS CE that is dual-homed to both PE1 and PE2 for redundant connectivity. However, CE4, which is also in the same VPLS domain, is single-homed to just PE1.

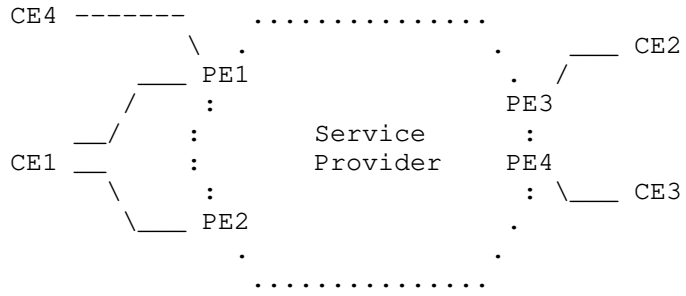


Figure 2: Scenario 2

2.2. VPLS Multi-homing Considerations

The first (perhaps obvious) fact about a multi-homed VPLS CE, such as CE1 in Figure 1 is that if CE1 is an Ethernet switch or bridge, a loop has been created in the customer VPLS. This is a dangerous situation for an Ethernet network, and the loop must be broken. Even if CE1 is a router, it will get duplicates every time a packet is flooded, which is clearly undesirable.

The next is that (unlike the case of IP-based multi-homing) only one of PE1 and PE2 can be actively sending traffic, either towards CE1 or into the SP cloud. That is to say, load balancing techniques will not work. All other PEs MUST choose the same designated forwarder

for a multi-homed site. Call the PE that is chosen to send traffic to/from CE1 the "designated forwarder".

In Figure 2, CE1 and CE4 must be dealt with independently, since CE1 is dual-homed, but CE4 is not.

3. Multi-homing Operation

This section describes procedures for electing a designated forwarder among the set of PEs that are multi-homed to a customer site. The procedures described in this section are applicable to BGP based VPLS, LDP based VPLS with BGP-AD or a VPLS that contains a mix of both BGP and LDP signaled PWs.

3.1. Customer Edge (CE) NLRI

Section 3.2.2 in [RFC4761] specifies a NLRI to be used for BGP based VPLS (BGP VPLS NLRI). The format of the BGP VPLS NLRI is shown below.

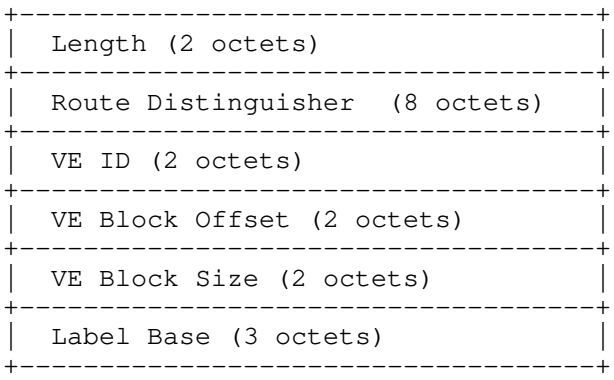


Figure 3: BGP VPLS NLRI

For multi-homing operation, a customer-edge NLRI (CE NLRI) is proposed that uses BGP VPLS NLRI with the following fields set to zero: VE Block Offset, VE Block Size and Label Base. In addition, the VE-ID field of the NLRI is set to CE-ID. Thus, the CE NLRI contains 2 octets indicating the length, 8 octets for Route Distinguisher, 2 octets for CE-ID and 7 octets with value zero.

It is valid to have non-zero VE block offset, VE block size and label base in the VPLS NLRI for a multi-homed site. VPLS operations, including multi-homing, in such a case are outside the scope of this document. However, for interoperability with existing deployments

that use non-zero VE block offset, VE block size and label base for multi-homing operation, Section 6.1 provides more detail.

Wherever VPLS NLRI is used in this document, context must be used to infer if it is applicable for CE NLRI, VE NLRI or for both.

3.2. Deployment Considerations

It is mandatory that each instance within a VPLS domain MUST be provisioned with a unique Route Distinguisher value. Unique Route Distinguisher allows VPLS advertisements from different VPLS PEs to be distinct even if the advertisements have the same VE-ID, which can occur in case of multi-homing. This allows standard BGP path selection rules to be applied to VPLS advertisements.

Each VPLS PE must advertise a unique VE-ID with non-zero VE Block Offset, VE Block Size and Label Base values in the BGP NLRI. VE-ID is associated with the base VPLS instance and the NLRI associated with it must be used for creating PWs among VPLS PEs. Any single-homed customer sites connected to the VPLS instance do not require any special addressing. However, an administrator (SP operator) can choose to have a CE-ID for a single-homed site as well. Any multi-homed customer sites connected to the VPLS instance require special addressing, which is achieved by use of CE-ID. A set of customer sites are distinguished as multi-homed if they all have the same CE-ID. The following examples illustrate the use of VE-ID and CE-ID.

Figure 1 shows a customer site, CE1, multi-homed to two VPLS PEs, PE1 and PE2. In order for all VPLS PEs to set up PWs to each other, each VPLS PE must be configured with a unique VE-ID for its base VPLS instance. In addition, in order for all VPLS PEs within the same VPLS domain to elect one of the multi-homed PEs as the designated forwarder, an indicator that the PEs are multi-homed to the same customer site is required. This is achieved by assigning the same VPLS site ID (CE-ID) on PE1 and PE2 for CE1. When remote VPLS PEs receive NLRI advertisement from PE1 and PE2 for CE1, the two NLRI advertisements for CE1 are identified as candidates for designated forwarder selection due to the same CE-ID. Thus, same CE-ID MUST be assigned on all VPLS PEs that are multi-homed to the same customer site.

Figure 2 shows two customer sites, CE1 and CE4, connected to PE1 with CE1 multi-homed to PE1 and PE2. Similar to Figure 1 provisioning model, each VPLS PE must be configured with a unique VE-ID for it base VPLS instance. CE1 which is multi-homed to PE1 and PE2 requires configuration of CE-ID and both PE1 and PE2 MUST be provisioned with the same CE-ID for CE1. CE2 and CE3 are single-homed sites and do not require special addressing. However, an operator must configure

a CE-ID for CE4 on PE1. By doing so, remote PEs can determine that PE1 has two VPLS sites, CE1 and CE4. If both CE1 and CE4 connectivity to PE1 is down, remote PEs can choose based on D bit in VE NLRI not to send multicast traffic to PE1 as there are no VPLS sites reachable via PE1. If CE4 was not assigned a unique CE-ID, remote PEs have no way to know if there are other VPLS sites attached and hence, would always send multicast traffic to PE1. While CE2 and CE3 can also be configured with unique CE-IDs, there is no advantage in doing so as both PE3 and PE4 have exactly one VPLS site.

Note that a CE-ID=0 is invalid and a PE should discard such an advertisement.

Use of multiple VE-IDs per VPLS instance for either multi-homing operation or for any other purpose is outside the scope of this document. However, for interoperability with existing deployments that use multiple VE-IDs, Section 6.1 provides more detail.

3.3. Designated Forwarder Election

BGP-based multi-homing for VPLS relies on standard BGP path selection and VPLS DF election. The net result of doing both BGP path selection and VPLS DF election is that of electing a single designated forwarder (DF) among the set of PEs to which a customer site is multi-homed. All the PEs that are elected as non-designated forwarders MUST keep their attachment circuit to the multi-homed CE in blocked status (no forwarding).

These election algorithms operate on VPLS advertisements, which include both the NLRI and attached BGP attributes. These election algorithms are applicable to all VPLS NLRIs, and not just to CE NLRIs. In order to simplify the explanation of these algorithms, we will use a number of variables derived from fields in the VPLS advertisement. These variables are: RD, SITE-ID, VBO, DOM, ACS, PREF and PE-ID. The notation ADV -> <RD, SITE-ID, VBO, DOM, ACS, PREF, PE-ID> means that from a received VPLS advertisement ADV, the respective variables were derived. The following sections describe two attributes needed for DF election, then describe the variables and how they are derived from fields in VPLS advertisement ADV, and finally describe how DF election is done.

3.3.1. Attributes

The procedures below refer to two attributes: the Route Origin community (see Section 4.1) and the L2-info community (see Section 4.2). These attributes are required for inter-AS operation; for generality, the procedures below show how they are to be used.

The procedures also outline how to handle the case that either or both are not present.

For BGP-based Multi-homing, ADV MUST contain an L2-info extended community as specified in [RFC4761]. Within this community are various control flags. Two new control flags are proposed in this document. Figure 4 shows the position of the new 'D' and 'F' flags.

Control Flags Bit Vector

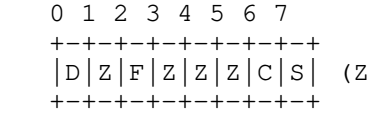


Figure 4

1. 'D' (Down): Indicates connectivity status. In case of CE NLRI, the connectivity status is between a CE site and a VPLS PE. In case of VE NLRI, the connectivity status is for the VPLS instance. In case of CE NLRI, the bit MUST be set to one if all the attachment circuits connecting a CE site to a VPLS PE are down. In case of VE NLRI, the bit must be set to one if the VPLS instance is operationally down. Note that a VPLS instance that has no connectivity to any of its sites must be considered as operationally down.

2. 'F' (Flush): Indicates when to flush MAC state. A designated forwarder must set the F bit and a non-designated forwarder must clear the F bit when sending BGP CE NLRIs for multi-homed sites. A state transition from one to zero for the F bit can be used by a remote PE to flush all the MACs learned from the PE that is transitioning from designated forwarder to non-designated forwarder. Refer to Section 5 for more details on the use case.

3.3.2. Variables Used

3.3.2.1. RD

RD is simply set to the Route Distinguisher field in the NLRI part of ADV. Actual process of assigning Route Distinguisher values must guarantee its uniqueness per PE node. Therefore, two multi-homed PEs offering the same VPLS service to a common set of CEs MUST allocate different RD values for this site respectively.

3.3.2.2. SITE-ID

SITE-ID is simply set to the VE-ID field in the NLRI part of the ADV.

Note that no distinction is made whether VE-ID is for a multi-homed site or not.

3.3.2.3. VBO

VBO is simply set to the VE Block Offset field in the NLRI part of ADV.

3.3.2.4. DOM

This variable, indicating the VPLS domain to which ADV belongs, is derived by applying BGP policy to the Route Target extended communities in ADV. The details of how this is done are outside the scope of this document.

3.3.2.5. ACS

ACS is the status of the attachment circuits for a given site of a VPLS. ACS = 1 if all attachment circuits for the site are down, and 0 otherwise.

ACS is set to the value of the 'D' bit in ADV that belongs to CE NLRI. If ADV belongs to base VPLS instance (VE NLRI) with non-zero label block values, no change must be made to ACS.

3.3.2.6. PREF

PREF is derived from the Local Preference (LP) attribute in ADV as well as the VPLS Preference field (VP) in the L2-info extended community. If the Local Preference attribute is missing, LP is set to 0; if the L2-info community is missing, VP is set to 0. The following table shows how PREF is computed from LP and VP.

VP Value	LP Value	PREF Value	Comment
0	0	0	malformed advertisement, unless ACS=1
0	1 to (2^16-1)	LP	backwards compatibility
0	2^16 to (2^32-1)	(2^16-1)	backwards compatibility
>0	LP same as VP	VP	Implementation supports VP
>0	LP != VP	0	malformed advertisement

Table 1

3.3.2.7. PE-ID

If ADV contains a Route Origin (RO) community (see Section 4.1) with type 0x01, then PE-ID is set to the Global Administrator sub-field of the RO. Otherwise, if ADV has an ORIGINATOR_ID attribute, then PE-ID is set to the ORIGINATOR_ID. Otherwise, PE-ID is set to the BGP Identifier.

3.3.3. Election Procedures

The election procedures described in this section apply equally to BGP VPLS and LDP VPLS. A distinction MUST NOT be made on whether the NLRI is a multi-homing NLRI or not. Subset of these procedures documented in standard BGP best path selection deals with general IP Prefix BGP route selection processing as defined in [RFC4271]. A separate part of the algorithm defined under VPLS DF election is specific to designated forwarded election procedures performed on VPLS advertisements. A concept of bucketization is introduced to define route selection rules for VPLS advertisements. Note that this is a conceptual description of the process; an implementation MAY choose to realize this differently as long as the semantics are preserved.

3.3.3.1. Bucketization for standard BGP path selection

An advertisement

ADV -> <RD, SITE-ID, VBO, ACS, PREF, PE-ID>

is put into the bucket for <RD, SITE-ID, VBO>. In other words, the information in BGP path selection consists of <RD, SITE-ID, VBO> and only advertisements with exact same <RD, SITE-ID, VBO> are candidates for BGP path selection procedure as defined in [RFC4271].

3.3.3.2. Bucketization for VPLS DF Election

An advertisement

ADV -> <RD, SITE-ID, VBO, DOM, ACS, PREF, PE-ID>

is discarded if DOM is not of interest to the VPLS PE. Otherwise, ADV is put into the bucket for <DOM, SITE-ID>. In other words, all advertisements for a particular VPLS domain that have the same SITE-ID are candidates for VPLS DF election.

3.3.3.3. Tie-breaking Rules

This section describes the tie-breaking rules for VPLS DF election. Tie-breaking rules for VPLS DF election are applied to candidate advertisements by all VPLS PEs and the actions taken by VPLS PEs based on the VPLS DF election result are described in Section 3.4.

Given two advertisements ADV1 and ADV2 from a given bucket, first compute the variables needed for DF election:

ADV1 -> <RD1, SITE-ID1, VBO1, DOM1, ACS1, PREF1, PE-ID1>
ADV2 -> <RD2, SITE-ID2, VBO2, DOM2, ACS2, PREF2, PE-ID2>

Note that SITE-ID1 = SITE-ID2 and DOM1 = DOM2, since ADV1 and ADV2 came from the same bucket. Then the following tie-breaking rules MUST be applied in the given order.

1. if (ACS1 != 1) AND (ACS2 == 1) ADV1 wins; stop
   if (ACS1 == 1) AND (ACS2 != 1) ADV2 wins; stop
   else continue

2. if (PREF1 > PREF2) ADV1 wins; stop;
   else if (PREF1 < PREF2) ADV2 wins; stop;
   else continue

3. if (PE-ID1 < PE-ID2) ADV1 wins; stop;
   else if (PE-ID1 > PE-ID2) ADV2 wins; stop;
   else ADV1 and ADV2 are from the same VPLS PE

If there is no winner and ADV1 and ADV2 are from the same PE, a VPLS PE MUST retain both ADV1 and ADV2.

3.4. DF Election on PEs

DF election algorithm MUST be run by all multi-homed VPLS PEs. In addition, all other PEs SHOULD also run the DF election algorithm. As a result of the DF election, multi-homed PEs that lose the DF election for a SITE-ID MUST put the ACs associated with the SITE-ID in non-forwarding state.

DF election result on the egress PEs can be used in traffic forwarding decision. Figure 2 shows two customer sites, CE1 and CE4, connected to PE1 with CE1 multi-homed to PE1 and PE2. If PE1 is the designated forwarder for CE1, based on the DF election result, PE3 can choose to not send unknown unicast and multicast traffic to PE2 as PE2 is not the designated forwarder for any customer site and it has no other single homed sites connected to it.

3.5. Pseudowire and Site-ID Binding Properties

For the use case where a single PE provides connectivity to a set of CEs from which some on multi-homed and others are not, only single pseudowire MAY be established. For example, if PE1 provides VPLS service to CE1 and CE4 which are both part of the same VPLS domain, but different sites, and CE1 is multi-homed, but CE4 is not (as described in figure 2), PE3 would establish only single pseudowire toward PE1. A design needs to ensure that regardless of PE1's forwarding state in respect to DF or non-DF for multi-homed CE1, PE3s access to CE4 is established. Since label allocation and pseudowire established is tied to site-ID, we need to ensure that proper pseudowire bindings are established.

For set of given advertisements with the common DOM but with different Site-ID values, a VPLS PE speaker SHOULD instantiate and bind the pseudowire based on advertisement with the lowest Site-ID value. Otherwise, binding would be completely random and during DF changes for multi-homed site, non-multi-homed CE might suffer traffic loss.

4. Multi-AS VPLS

This section describes multi-homing in an inter-AS context.

4.1. Route Origin Extended Community

Due to lack of information about the PEs that originate the VPLS NLRIs in inter-AS operations, Route Origin Extended Community [RFC4360] is used to carry the source PE's IP address.

To use Route Origin Extended Community for carrying the originator VPLS PE's loopback address, the type field of the community MUST be set to 0x01 and the Global Administrator sub-field MUST be set to the PE's loopback IP address.

4.2. VPLS Preference

When multiple PEs are assigned the same site ID for multi-homing, it is often desired to be able to control the selection of a particular PE as the designated forwarder. Section 3.5 in [RFC4761] describes the use of BGP Local Preference in path selection to choose a particular NLRI, where Local Preference indicates the degree of preference for a particular VE. The use of Local Preference is inadequate when VPLS PEs are spread across multiple ASes as Local Preference is not carried across AS boundary. A new field, VPLS preference (VP), is introduced in this document that can be used to accomplish this. VPLS preference indicates a degree of preference for a particular customer site. VPLS preference is not mandatory for intra-AS operation; the algorithm explained in Section 3.3 will work with or without the presence of VPLS preference.

Section 3.2.4 in [RFC4761] describes the Layer2 Info Extended Community that carries control information about the pseudowires. The last two octets that were reserved now carries VPLS preference as shown in Figure 5.

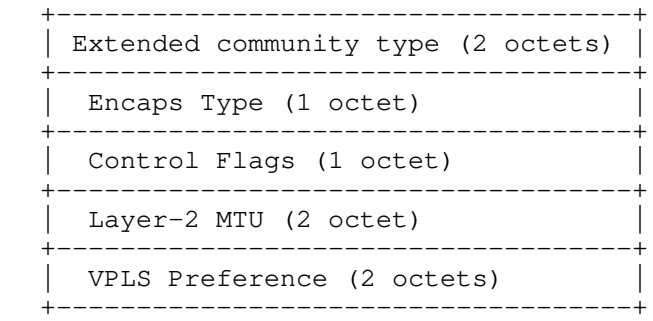


Figure 5: Layer2 Info Extended Community

A VPLS preference is a 2-octets unsigned integer. A value of zero indicates absence of a VP and is not a valid preference value. This interpretation is required for backwards compatibility. Implementations using Layer2 Info Extended Community as described in (Section 3.2.4) [RFC4761] MUST set the last two octets as zero since it was a reserved field.

For backwards compatibility, if VPLS preference is used, then BGP Local Preference MUST be set to the value of VPLS preference. Note that a Local Preference value of zero for a CE-ID is not valid unless 'D' bit in the control flags is set (see [I-D.kothari-l2vpn-auto-site-id]). In addition, Local Preference value greater than or equal to 2^16 for VPLS advertisements is not valid.

4.3. Use of BGP attributes in Inter-AS Methods

Section 3.4 in [RFC4761] and section 4 in [RFC6074] describe three methods (a, b and c) to connect sites in a VPLS to PEs that are across multiple AS. Since VPLS advertisements in method (a) do not cross AS boundaries, multi-homing operations for method (a) remain exactly the same as they are within as AS. However, for method (b) and (c), VPLS advertisements do cross AS boundary. This section describes the VPLS operations for method (b) and method (c). Consider Figure 6 for inter-AS VPLS with multi-homed customer sites.

4.3.1. Inter-AS Method (b): EBGP Redistribution of VPLS Information between ASBRs

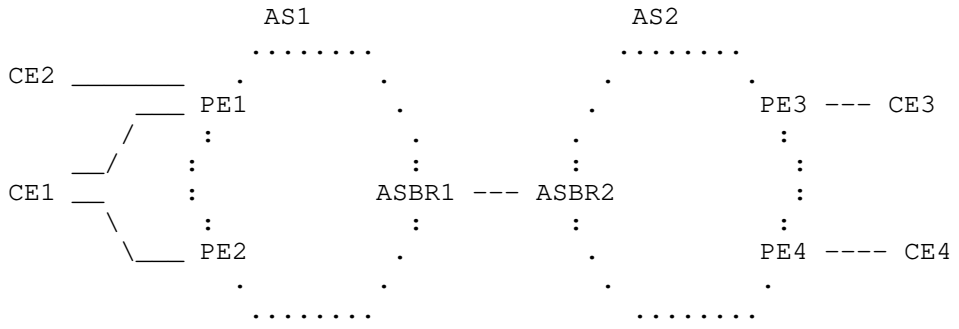


Figure 6: Inter-AS VPLS

A customer has four sites, CE1, CE2, CE3 and CE4. CE1 is multi-homed to PE1 and PE2 in AS1. CE2 is single-homed to PE1. CE3 and CE4 are also single homed to PE3 and PE4 respectively in AS2. Assume that in addition to the base LDP/BGP VPLS addressing (VSI-IDs/VE-IDs), CE-ID 1 is assigned for CE1. After running DF election algorithm, all four VPLS PEs must elect the same designated forwarder for CE1 site. Since BGP Local Preference is not carried across AS boundary, VPLS

preference as described in Section 4.2 MUST be used for carrying site preference in inter-AS VPLS operations.

For Inter-AS method (b) ASBR1 will send a VPLS NLRI received from PE1 to ASBR2 with itself as the BGP nexthop. ASBR2 will send the received NLRI from ASBR1 to PE3 and PE4 with itself as the BGP nexthop. Since VPLS PEs use BGP Local Preference in DF election, for backwards compatibility, ASBR2 MUST set the Local Preference value in the VPLS advertisements it sends to PE3 and PE4 to the VPLS preference value contained in the VPLS advertisement it receives from ASBR1. ASBR1 MUST do the same for the NLRIs it sends to PE1 and PE2. If ASBR1 receives a VPLS advertisement without a valid VPLS preference from a PE within its AS, then ASBR1 MUST set the VPLS preference in the advertisements to the Local Preference value before sending it to ASBR2. Similarly, ASBR2 must do the same for advertisements without VPLS Preference it receives from PEs within its AS. Thus, in method (b), ASBRs MUST update the VPLS and Local Preference based on the advertisements they receive either from an ASBR or a PE within their AS.

In Figure 6, PE1 will send the VPLS advertisements with Route Origin Extended Community containing its loopback address. PE2 will do the same. Even though PE3 receives the VPLS advertisements for VE-ID 1 and 2 from the same BGP nexthop, ASBR2, the source PE address contained in the Route Origin Extended Community is different for the CE1 and CE2 advertisements, and thus, PE3 creates two PWs, one for CE1 (for VE-ID 1) and another one for CE2 (for VE-ID 2).

4.3.2. Inter-AS Method (c): Multi-Hop EBGP Redistribution of VPLS Information between ASes

In this method, there is a multi-hop E-BGP peering between the PEs or Route Reflectors in AS1 and the PEs or Route Reflectors in AS2. There is no VPLS state in either control or data plane on the ASBRs. The multi-homing operations on the PEs in this method are exactly the same as they are in intra-AS scenario. However, since Local Preference is not carried across AS boundary, the translation of LP to VP and vice versa MUST be done by RR, if RR is used to reflect VPLS advertisements to other ASes. This is exactly the same as what a ASBR does in case of method (b). A RR must set the VP to the LP value in an advertisement before sending it to other ASes and must set the LP to the VP value in an advertisement that it receives from other ASes before sending to the PEs within the AS.

5. MAC Flush Operations

In a service provider VPLS network, customer MAC learning is confined to PE devices and any intermediate nodes, such as a Route Reflector, do not have any state for MAC addresses.

Topology changes either in the service provider's network or in customer's network can result in the movement of MAC addresses from one PE device to another. Such events can result into traffic being dropped due to stale state of MAC addresses on the PE devices. Age out timers that clear the stale state will resume the traffic forwarding, but age out timers are typically in minutes, and convergence of the order of minutes can severely impact customer's service. To handle such events and expedite convergence of traffic, flushing of affected MAC addresses is highly desirable.

5.1. MAC Flush Indicators

If 'D' bit in the control flags is set in a received VE NLRI, the receiving PE SHOULD flush all the MAC addresses learned from the PE advertising the failure.

Anytime a designated forwarder change occurs, a remote PE SHOULD flush all the MAC addresses it learned from the PE that lost the DF election (old designated forwarder). If multiple customer sites are connected to the same PE, PE1 as shown in Figure 2, and redundancy per site is desired when multi-homing procedures described in this document are in effect, then it is desirable to flush just the relevant MAC addresses from a particular site when the site connectivity is lost. However, procedures for flushing a limited set of MAC addresses are beyond the scope of this document. Use of either 'D' or 'F' bit in control flags only allows to flush all MAC addresses associated with a PE.

Designated forwarder change can occur in absence of failures, such as when an attachment circuit comes up. Consider the case in Figure 2 where PE1-CE1 link is non-operational and PE2 is the designated forwarder for CE1. Also assume that Local Preference of PE1 is higher than PE2. When PE1-CE1 link becomes operational, PE1 will send a BGP CE advertisement for CE1 to all it's peers. If PE3 performs the DF election before PE2, there is a chance that PE3 might learn MAC addresses from PE2 after it was done electing PE1. This can happen since PE2 has not yet processed the BGP CE advertisement from PE1 and as a result continues to send traffic to PE3. This can cause traffic from PE3 to CE1 to black-hole until those MAC addresses are deleted due to age out timers. Therefore, to avoid such race-conditions, a designated forwarder must set the F bit and a non-designated forwarder must clear the F bit when sending BGP CE

advertisements. A state transition from one to zero for the 'F' bit can be used by a remote PE to flush all the MACs learned from the PE that is transitioning from designated forwarder to non-designated forwarder.

5.2. Minimizing the effects of fast link transitions

Certain failure scenarios may result in fast transitions of the link towards the multi-homing CE which in turn will generate fast status transitions of one or multiple multi-homed sites reflected through multiple BGP CE advertisements and LDP MAC Flush messages.

It is recommended that a timer to damp the link flaps be used for the port towards the multi-homed CE to minimize the number of MAC Flush events in the remote PEs and the occurrences of BGP state compression for F bit transitions. A timer value more than the time it takes BGP to converge in the network is recommended.

6. Backwards Compatibility

No forwarding loops are formed when PEs or Route Reflectors that do not support procedures defined in this section co exist in the network with PEs or Route Reflectors that do support.

6.1. BGP based VPLS

As explained in this section, multi-homed PEs to the same customer site MUST assign the same CE-ID and related NLRI SHOULD contain the block offset, block size and label base as zero. Remote PEs that lack support of multi-homing operations specified in this document will fail to create any PWs for the multi-homed CE-IDs due to the label value of zero and thus, the multi-homing NLRI should have no impact on the operation of Remote PEs that lack support of multi-homing operations specified in this document.

For compatibility with PEs that use multiple VE-IDs with non-zero label block values for multi-homing operation, it is a requirement that a PE receiving such advertisements must use the labels in the NLRIs associated with lowest VE-ID for PW creation. It is possible that maintaining PW association with lowest VE-ID can result in PW flap, and thus, traffic loss. However, it is necessary to maintain the association of PW with the lowest VE-ID as it provides deterministic DF election among all the VPLS PEs.

6.2. LDP VPLS with BGP Auto-discovery

The BGP-AD NLRI has a prefix length of 12 containing only a 8 bytes RD and a 4 bytes VSI-ID. If a LDP VPLS PEs running BGP AD lacks support of multi-homing operations specified in this document, it SHOULD ignore a CE NLRI with the length field of 17. As a result it will not ask LDP to create any PWs for the multi-homed Site-ID and thus, the multi-homing NLRI should have no impact on LDP VPLS operation. MH PEs may use existing LDP MAC Flush to flush the remote LDP VPLS PEs or may use the MAC Flush procedures as described in Section 5

7. Security Considerations

No new security issues are introduced beyond those that are described in [RFC4761] and [RFC4762].

8. IANA Considerations

IANA already has a registry for "Layer2 Info Extended Community Control Flags Bit Vector" <https://www.iana.org/assignments/bgp-extended-communities>

This document requires two new bit flags to be assigned as follows:

Value	Name	Reference
D	Down connectivity status	This document
F	MAC flush indicator	This document

9. Contributing Authors

The authors would also like to thank Senad Palislamovic and Wen Lin for their contribution to the development of this document.

Senad Palislamovic
Nokia
Email: senad.palislamovic@nokia.com

Wen Lin
Juniper Networks
Email: wlin@juniper.net

10. Acknowledgments

The authors would like to thank Yakov Rekhter, Nischal Sheth, Mitali Singh, Ian Cowburn and Jonathan Hardwick for their insightful comments and probing questions.

11. References

11.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[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, <https://www.rfc-editor.org/info/rfc4761>.

[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, <https://www.rfc-editor.org/info/rfc6074>.

11.2. Informative References

[I-D.kothari-l2vpn-auto-site-id] Kothari, B., Kompella, K., and T. IV, "Automatic Generation of Site IDs for Virtual Private LAN Service", draft-kothari-l2vpn-auto-site-id-01 (work in progress), October 2008.

[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, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

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

[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, <https://www.rfc-editor.org/info/rfc4762>.

Authors' Addresses

Bhupesh Kothari
Augtera Networks

Email: bhupesh@anvaya.net

Kireeti Kompella
Juniper Networks
1194 N. Mathilda Ave.
Sunnyvale, CA 94089
US

Email: kireeti.kompella@gmail.com

Wim Henderickx
Nokia

Email: wim.henderickx@nokia.com

Florin Balus
Cisco

Email: fbalus@gmail.com

James Uttaro
AT&T
200 S. Laurel Avenue
Middletown, NJ 07748
US

Email: uttaro@att.com

Service Function Chaining
Internet-Draft
Intended status: Informational
Expires: April 17, 2015

S. Mackie
B. Rijsman
Juniper Networks
M. Napierala
AT&T
D. Daino
Telecom Italia
D.R. Lopez
Telefonica I+D
D. Bernier
Bell Canada
W. Haeffner
Vodafone

October 17, 2014

Service Function Chains Using Virtual Networking
draft-mackie-sfc-using-virtual-networking-02.txt

Status of this Memo

Copyright Notice

Abstract

This document describes how service function chains (SFC) can be applied to traffic flows using routing in a virtual (overlay) network to steer traffic between service nodes. Chains can include services running in routers, on physical appliances or in virtual machines. Service chains have applicability at the subscriber edge, business edge and in multi-tenant datacenters. The routing function into SFCs and between service functions within an SFC can be performed by physical devices (routers), be virtualized inside hypervisors, or run as part of a host OS.

The architecture uses a controller to calculate and install routes to implement an SFC, based on a topological model of the chain and knowledge of the network addresses that should pass through the chain. An advantage of the approach is that SFCs can be implemented without alteration to today's BGP standard, and without change to the current operation of routers.

Service chains need to support load balancing between network functions, and symmetric forward and reverse paths are required when stateful services are involved. This document shows how these requirements can be met by using VRFs at the ingress and egress of each service instance and by performing load balancing after the egress of each service as part of the routing function.

Table of Contents

1. Introduction

The purpose of networks is to allow computing systems to communicate with each other. Traditionally, requests are made from the client or customer side of a network, and responses are generated by applications residing in a datacenter. Over time, the network between the client and the application has become more complex, and traffic between the client and the application is acted on by intermediate systems that apply network services. Some of these activities, like firewall filtering, subscriber attachment and network address translation are generally carried out in network devices along the traffic path, while others are carried out by dedicated appliances, such as media proxy and deep packet inspection (DPI). Deployment of these in-network services is complex, time-consuming and costly, since they require configuration of devices with vendor-specific operating systems, sometimes with co-processing cards, or deployment of physical devices in the network, which requires cabling and configuration of the devices that they connect to. Additionally, other devices in the network itself need to be configured to ensure that traffic is correctly steered through the systems that services are running on.

The current mode of operations does not easily allow common operational processes to be applied to the lifecycle of services in the network, or for steering of traffic through them.

The recent emergence of Network Functions Virtualization (NFV) [NFVE2E] to provide a standard deployment model for network services as software appliances, combined with Software Defined Networking (SDN) for more dynamic traffic steering can provide foundational elements that will allow network services to be deployed and managed far more efficiently and with more agility than is possible today.

This document describes how the combination of several existing technologies can be used to create chains of functions, while preserving the requirements of scale, performance and reliability for service provider networks. The technologies employed are:

- Traffic flow between service functions described by routing and network policies rather than by static physical or logical connectivity
- Packet header encapsulation in order to create virtual private networks using network overlays
- VRFs on both physical devices and in hypervisors to implement forwarding policies that are specific to virtual networks

- Use of a controller to calculate routes to be installed in routing systems to form a service chain. The controller uses a topological model that stores service function instance connectivity to network devices and intended connectivity between service functions.

- MPLS or other labeling to facilitate identification of the next interface to send packets to in a service function chain

- BGP or BGP-style signaling to distribute routes in order to create service function chains

- Distributed load balancing between service functions performed in the VRFs that service function instance connect to.

When BGP signaling and MPLS labeling are used, service function chains can be introduced to today's networks, using existing network equipment and avoiding the need to introduce new network protocols, modify existing protocols or develop device enhancements.

Virtualized environments can be supported without running BGP or MPLS natively in data centers by encapsulating BGP messages in a more generally deployed protocol, such as XMPP, and by using GRE or VXLAN encapsulation for transport.

Traffic can be directed into service function chains using IP routing at each end of the service function chain, or be directed into the chain by a classifier.

The architecture can support an evolution from services implemented in physical devices attached to physical forwarding systems (routers) to fully virtualized implementations as well as intermediate hybrid implementations.

1.1. Terminology

This document follows some of the terminology used in [draft-quinn-sfc-arch] and adds some new terminology:

Network Service: An externally visible service offered by a network operator; a service may consist of a single service function or a composite built from several service functions executed in one or more pre-determined sequences and delivered by software executing in physical or virtual devices.

Classification: Locally applied customer/network/service policy used to identify and select traffic flow(s) requiring appropriate outbound forwarding actions, in particular, to direct specific

traffic flows into the ingress of a particular service function chain, or causing branching within a service function chain.

Virtual Network: A logical overlay network built via virtual links or packet encapsulation, over an existing network (the underlay).

Service Function Chain (SFC): A service function chain defines an ordered set of service functions that must be applied to packets and/or frames selected as a result of classification. AN SFC may be either a linear chain or a complex service graph with multiple branches.

SFC Set: The pair of SFCs through which the forward and reverse directions of a given classified flow will pass.

Service Function (SF): A function that is responsible for specific treatment of received packets. A Service Function can act at the network layer or other OSI layers. A Service Function can be embedded in one or more physical network elements, or can be implemented in one or more software instances running on physical or virtual hosts. One or multiple Service Functions can be embedded in the same network element or run on the same host. Multiple instances of a Service Function can be enabled in the same administrative domain. We will also refer to "Service Function" as, simply, "Service" for simplicity.

A non-exhaustive list of Services includes: firewalls, DDOS protection, anti-malware/ant-virus systems, WAN and application acceleration, Deep Packet Inspection (DPI), server load balancers, network address translation, HTTP Header Enrichment functions, video optimization, TCP optimization, etc.

SF Instance: An instance of software that implements the packet processing of a service function

SF Instance Set: A group of SF instances that, in parallel, implement a service function in an SFC.

Routing System: A hardware or software system that performs layer 3 routing and/or forwarding functions. The term includes physical routers as well as hypervisor or Host OS implementations of the forwarding plane of a conventional router.

VRF: A subsystem within a routing system as defined in [RFC4364] that contains private routing and forwarding tables and has physical and/or logical interfaces associated with it. In the case of hypervisor/Host OS implementations, the term refers only

to the forwarding function of a VRF, and this will be referred to as a "VPN forwarder."

Ingress VRF: A VRF containing an ingress interface of a SF instance

Egress VRF: A VRF containing an egress interface of a SF instance

2. Service Function Chain Architecture Using Virtual Networking

The architecture described in this document uses virtual networks managed with a controller to implement service function chains. Service function chains can be implemented on devices that support today's MPLS VPN and BGP standards [RFC4364, RFC4271, RFC4760], without requiring additional feature development, but other encapsulations, such as VXLAN [draft-mahalingam-vxlan], could be used, and use of other control plane protocols is possible.

The following sections detail the building blocks of the architecture, and outlines the process of route installation by the controller to create an SFC. A detailed, worked example of SFC instantiation that includes routing table and packet flow details is contained in Appendix A.

2.1. High Level Architecture

Service function chains can be deployed with or without a classifier. Use cases where SFCs may be deployed without a classifier include multi-tenant data centers, private and public cloud and virtual CPE for business services. Classifiers will primarily be used in mobile and wireline subscriber edge use cases. Use of a classifier is discussed in Section .

A high-level architecture diagram of an SFC without a classifier, where traffic is routed into and out of the SFC, is shown in Figure 1, below.

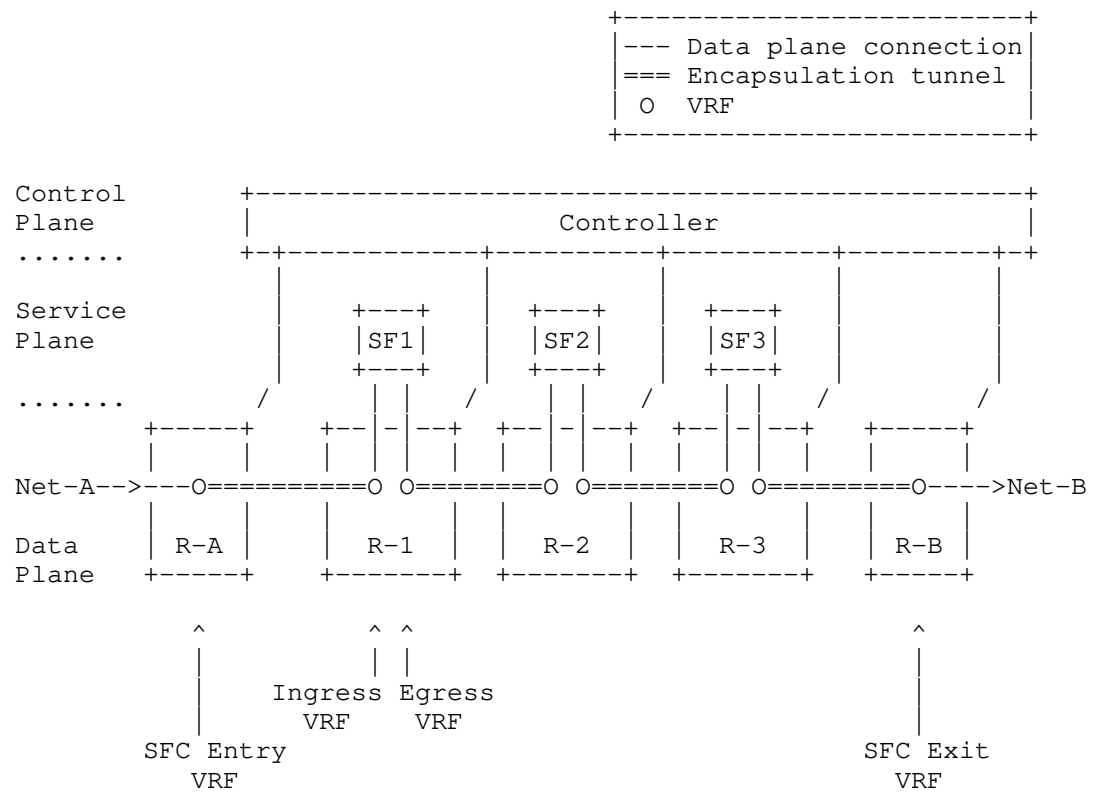


Figure 1- High level SFC Architecture

The diagram shows an SFC that traffic from Network-A destined for Network-B will pass through. Routing system R-A contains a VRF (shown as "O" symbol) that is the SFC entry point. This VRF will advertise a route to Network-B into Network-A causing any traffic from a source in Network-A with a destination in Network-B to arrive in this VRF. The forwarding table in the VRF in R-A will direct traffic destined for Network-B into an encapsulation tunnel with destination R-1 and a label that identifies the ingress (left) interface of SF1 that R-1 should send the packets out on. The packets are processed by service instance SF-1 and arrive in the egress (right) VRF in R-1. The forwarding entries in the egress VRF direct traffic to the next ingress VRF using encapsulation tunneling. The process is repeated for each service instance in the SFC until packets arrive at the SFC exit VRF (in R-B). This VRF is peered with Network-B and routes packets towards their destinations in the user data plane.

In the example, each pair of ingress and egress VRFs are configured in separate routing systems, but such pairs could be collocated in the same routing system, and it is possible for the ingress and egress VRFs for a given SF instance to be in different routing systems. The SFC entry and exit VRFs can be collocated in the same routing system, and the service instances can be local or remote from either or both of the routing systems containing the entry and exit VRFs, and from each other.

The controller is responsible for configuring the VRFs in each routing system, installing the routes in each of the VRFs to implement the SFC, and, in the case of virtualized services, may instantiate the service instances.

2.2. Summary of Operation

The controller installs forwarding entries or distributes routes to each ingress VRF to direct traffic into the ingress interfaces of the service instances that will form an SFC. Overlay networking is used to transport traffic between the egress VRF of one service instance and the ingress VRF of the next.

Traffic may be directed into an SFC via routing as shown in the diagram, or a classifier may be used to select an SFC entry point based on filter criteria (see Section).

The controller, which functions as an extended BGP route reflector, exchanges routes between routing systems, which can be physical devices (conventional router) or be running as the forwarding function of a hypervisor or Host OS (VPN forwarder). The controller connects service instances of different services to each other by creating VPNs based on route targets in advertised routes. The controller directs traffic into SF instance ingress interfaces by installing particular routes based on the controller’s knowledge of connectivity of service instances to VRFs, and the desired SFC topology. In most cases, a controller will configure forward and reverse SFCs at the same time, and this will result in routes being installed in the VRFs on both sides of each SF instance in a symmetrical SFC set.

As will be described in detail in this document, the architecture allows the logical connection between two service functions to be described by a virtual private network, and to be implemented using well-known MPLS L3 VPN technology. The forwarding into service function instances, and between service function instances, is performed by VRFs, which provide convenient containers within which to specify traffic forwarding polices, such as load balancing, and QoS shaping and policing. The controller maintains a global SFC

topological model, and the VRFs are configured only with next hops to locally connected service instances, and to other routing systems that are connected to instances of the next service in the SFC.

This architecture does not rely on any new metadata header to be carried with the user packets for steering in an SFC [draft-boucadair-sfc-arch, draft-quinn-sfc-nsh], although metadata for use inside service functions would be supported [draft-rijsman-sfc-metadata-considerations].

2.3. Service Function Chain Logical Model

A service function chain is a set of logically connected service functions through which traffic can flow. Each egress interface of one service function is logically connected to an ingress interface of the next service function.

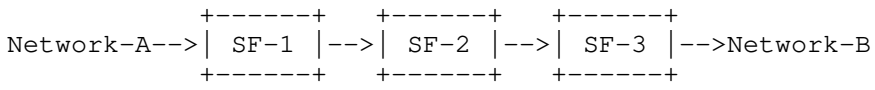


Figure 2- A Chain of Service Functions

In Figure 2, above, a service function chain has been created that connects Network-A to Network-B, such that traffic from a host in Network-A to a host in Network-B will traverse the service function chain.

As defined in [draft-boucadair-sfc-arch], a service function chain is uni-directional, while in [draft-quinn-sfc-arch] SFCs can be unidirectional or bi-directional. In this document, in order to allow for the possibility that the forward and reverse paths may not be symmetrical, SFCs are defined as uni-directional, and the term "SFC set" is used to refer to a pair of forward and reverse direction SFCs for some set of routed or classified traffic.

2.4. Service Function Implemented in a Set of SF Instances

A service function instance is a software system that acts on packets that arrive on an ingress interface of that software system. Service function instances may run on a physical appliance or in a virtual machine. A service function instance may be transparent at layer 2 and/or 3, and may support branching across multiple egress interfaces and/or aggregation across ingress interfaces. For simplicity, the examples in this document have a single ingress and a single egress interface.

Each service function in a chain can be implemented by a single service function instance, or by a set of instances in order to provide scale and resilience.

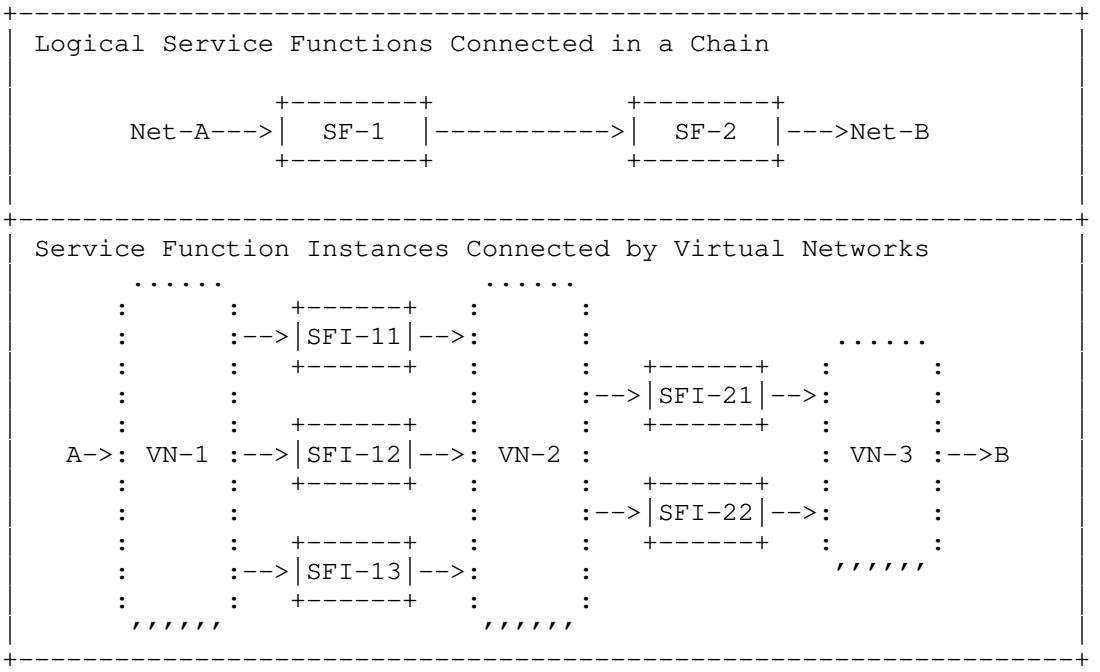


Figure 3- Service Functions Are Composed of SF Instances Connected Via Virtual Networks

In Figure 3, service function SF-1 is implemented in three service function instances, SFI-11, SFI-12, and SFI-13. Service function SF-2 is implemented in two SF instances. The service function instances are connected to the next service function in the chain using a virtual network, VN-2. Additionally, a virtual network (VN-1) is used to enter the SFC and another (VN-3) is used at the exit.

The logical connection between two service functions is implemented using a virtual network that contains egress interfaces for instances of one service function, and ingress interfaces of instances of the next service function. Traffic is directed across the virtual network between the two sets of service function instances using layer 3 forwarding (MPLS VPN).

The virtual networks could be described as "directed half-mesh", in that the egress interface of each SF instance of one service

function can reach any ingress interface of the SF instances of the connected service function.

Details on how routing across virtual networks is achieved, and requirements on load balancing across ingress interfaces are discussed in later sections of this document.

2.5. SF Instance Connections to VRFs

SF instances can be deployed as software running on physical appliances, or in virtual machines running on a hypervisor.

2.5.1. SF Instance in Physical Appliance

The case of a SF instance running on a physical appliance is shown in Figure 4, below.

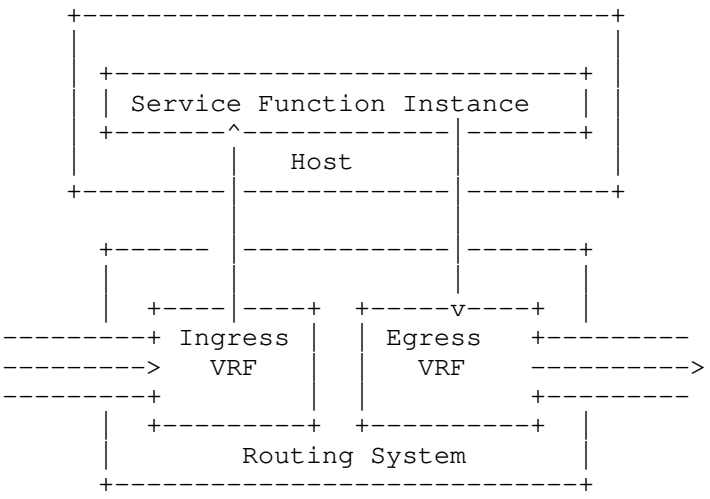


Figure 4- Ingress and Egress VRFs for a Physical Routing System and Physical SF Instance

The routing system is a physical device and the service function instance is implemented as software running in a physical appliance (host) connected to it. Transport between VRFs on different routing systems that are connected to other SF instances in an SFC is via encapsulation tunnels, such as MPLS over GRE.

2.5.2. SF Instance in a Virtualized Environment

In virtualized environments, a routing system with VRFs that act as VPN forwarders is resident in the hypervisor/Host OS, and is co-

resident in the host with one or more SF instances that run in virtual machines. The egress VPN forwarder performs tunnel encapsulation to send packets to other physical or virtual routing systems with attached SF instances to form an SFC. The tunneled packets are sent through the physical interfaces of the host to the other hosts or physical routers. This is illustrated in Figure 5, below.

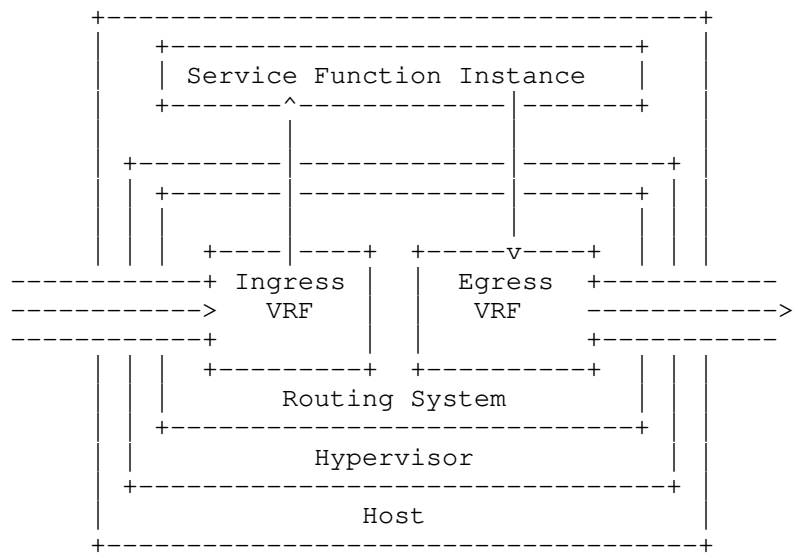


Figure 5- Ingress and Egress VRFs for a Virtual Routing System and Virtualized SF Instance

When more than one SF instance is running on a hypervisor, they can be connected to the same VRF for scale out of an SF within an SFC, or to different VRFs if the SF instances implement different service functions in the same SFC or when they belong to different SFCs.

The routing mechanisms in the VRFs into and between service function instances, and the encapsulation tunneling between routing systems are identical in the physical and virtual implementation of SFCs described in this document. Physical and virtual service functions can be mixed as needed with different combinations of physical and virtual routing systems.

2.6. Encapsulation Tunneling for Transport

Encapsulation tunneling is used to transport packets between SF instances in the chain and, when a classifier is not used, from the originating network into the SFC and from the SFC into the destination network. Tunneling is enabled between all systems that have VRFs that share route targets.

The tunnels can be MPLS over GRE [RFC4023], MPLS over UDP [draft-ietf-mpls-in-udp], MPLS over MPLS [RFC3031], VXLAN [draft-mahalingam-vxlan], or another suitable encapsulation method.

Tunneling may be enabled in each routing system as part of a base configuration, or may be configured by the controller.

2.7. Controller Function

The purpose of the controller is to manage instantiation of SFCs in networks and datacenters. When an SFC is to be instantiated, a model of the desired topology (service functions, number of instances, connectivity) is built in the controller either via an API or GUI. The controller then selects resources in the infrastructure that will support the SFC and configures them. This can involve instantiation of SF instances to implement each service function, the instantiation of VRFs that will form virtual networks between SF instances, and installation of routes to cause traffic to flow into and between SF instances.

For simplicity, in this document, the controller is assumed to contain all the required features for management of SFCs. In actual implementations, these features may be distributed among multiple inter-connected systems. E.g. An overarching orchestrator might manage the overall SFC model, sending instructions to a separate virtual machine manager to instantiate service function instances, and to a virtual network manager to set up the service chain connections between them.

2.7.1. Controller for SFC with Physical Routers

In physical devices, a configuration interface (e.g. Netconf [RFC6241]) is used to create and configure VRFs, while BGP signaling is used to advertise route updates.

When physical routers that support BGP are used to connect SF instances in an SFC, the controller acts as a conventional BGP route-reflector with BGP sessions to each routing system. The controller has the additional capability to install routes whose next hop is a local interface name. These routes are needed to

enable traffic to flow into the SF instances. These routes cannot be installed using BGP, and here it is assumed that Netconf will be used to make these changes. The routes are calculated based on the topological model of the desired SFC which is mapped onto specific SF instances whose connectivity to VRFs is also known by the controller.

2.7.2. SFC Implementation Procedure with Physical Routers

The following is a summary of the steps involved in creating an SFC using physical routers that fully support BGP. The service function instances are assumed to be already connected to logical interfaces on the router. A linear service chain is assumed in this example with each service function implemented in SF instances each with two interfaces.

A request to create a new SFC is made using an API or GUI. The request will include the functions in the chain, their order along the chain, and indication of the scale requirement for each function. The request will also identify the networks connected to each end of the SFC. The number of instances of each function may be calculated by the controller based on capacity or performance requirements, or may be specified in the creation request.

The service instances to be used in the SFC may be specified in the creation request, or may be selected by the controller from a set of available service instances.

The process of SFC creation is as follows:

1. Controller creates a VRF (via Netconf) in each router that is connected to a service instance that will be used in the SFC

2. Controller configures each VRF (via Netconf) to contain the logical interface that connects to a SF instance.

3. Controller implements route target import and export policies in the VRFs (via Netconf) using the same route targets for ingress and egress VRFs of connected services in the SFC.

4. Controller installs (via Netconf) a static route in each ingress VRF whose next hop is the interface that a SF instance is connected to. The prefix for the route is the destination network to be reached by passing through the SFC.

5. Routing systems advertise the static routes back to the controller (via BGP) as VPN routes with next hop being the IP address of the router, with an encapsulation specified and a label that identifies the service instance interface.

6. Controller sends route updates (via BGP) to all routers containing VRFs with matching route targets.

7. Routes are installed in egress VRFs with matching import targets. The egress VRFs of each SF instance will now contain VPN routes to one or more routers containing ingress VRFs for SF instances of the next service function in the SFC.

Traffic entering a VRF at one end of the SFC will be directed sequentially through a set of SF instances that implement each service function in the SFC.

If multiple SF instances are deployed for a given service function, the VRFs at the egress of the previous service function will load balance across the ingress interfaces of the SF instances of the next service function. Load balancing is described in Section .

The forward and reverse direction SFCs in an SFC set may be configured at the same time in the above procedure. For a symmetric SFC set, each VRF will be an ingress VRF in one direction, and an egress VRF in the other direction.

A detailed example is described in Appendix A.

2.7.3. Controller for SFC with Virtualized Routing

The process of VRF creation and route exchange is different for virtualized routing systems that implement only a VPN forwarding function and do not support a full BGP implementation.

For example, in [draft-ietf-l3vpn-end-system], the actions of BGP and Netconf are performed using XMPP in a virtualized environment. The content of the XMPP messages from the controller corresponds to configuration commands in Netconf and to route update messages in BGP when physical devices are used.

The controller of [draft-ietf-l3vpn-end-system] has a global model of all the routing systems under its control, including VRFs, interfaces, and route targets. The controller calculates all required routes based on dynamically assigned route targets and on import/export policies, and sends instructions to hypervisors/Host OS via XMPP to program the forwarding plane. The only route updates

sent from a hypervisor/Host OS to the controller occur when a virtual machine is created, destroyed, or failed.

In [draft-ietf-l3vpn-end-system] the controller is able to peer via BGP with routers and route reflectors for route exchange with physical networks.

2.7.4. SFC Implementation Procedure with Virtual Routers

The following is a summary of the steps involved in creating an SFC using this architecture when VPN forwarders are used in hypervisors or Host OS's, and the controller implements internal BGP emulation.

A request to create a new SFC is made using an API or GUI. The request will include the functions in the chain, their order along the chain, and indication of the scale requirement for each function. The request will also identify the networks connected to each end of the SFC. The number instances of each function may be calculated by the controller based on capacity or performance requirements or specified by the in the creation request.

The service instances to be used in the SFC may be specified in the creation request, or may be created, as needed, by the controller on suitable hosts from a set of available images.

The process of SFC creation is as follows

1. The controller creates service function instances as virtual machines running on hypervisors. The placement of each SF instance on a specific server can be according to an algorithm that takes into account geography, server type, data center environment (e.g. power supply, rack location) and other criteria, but details are out of scope for this document.

2. Controller creates an internal model of VRFs that each SF instance will connect to.

3. Controller adds the hypervisor interfaces that SF instances are connected to into each VRF in its internal model.

4. Controller instantiates the route target import and export policies in the VRFs in its internal model.

5. Controller, in its internal model, installs a static route in each ingress VRF whose next hop is the interface that a SF instance is connected to. The prefix for the route is the network that will ultimately be reached by passing through the SF instance.

6. Controller calculates the effect of route reflection for the static routes in its model, which results in the controller installing routes in modeled VRFs with matching route targets. These routes have next hops pointing to the VRFs containing static routes. These new routes will direct traffic from the egress of one SF instance to the ingress of the next when installed into the actual VRFs in the infrastructure. In a hybrid physical/virtual SFC the controller would also perform actual route reflection with peered routers that host ingress and egress VRFs of SFCs.

7. Controller creates actual VRFs in the virtual routing systems in hypervisors/Host OS where SF instances reside.

8. The static and reflected routes that were calculated in the controller are sent to the VRFs and installed in their forwarding tables. The egress VRF for each SF instance will now contain VPN routes to one or more systems with ingress VRFs of SF instances of the next service function in the SFC.

Traffic entering a VRF at one end of the SFC will be directed sequentially through a set of SF instances that implement each service function in the SFC.

If multiple SF instances are deployed for a given service function, the VRFs at the egress of the previous service function will load balance across the ingress interfaces of the SF instances of the next service function. Load balancing is described in Section .

2.8. A Variation on Setting Prefixes in an SFC

In the configuration method described above, the network prefixes for each network (Network-A and Network-B in the example above) connected to the SFC are used in the routes that direct traffic through the SFC. This creates an operational linkage between the implementation of the SFC and the insertion of the SFC into a network.

For instance, subscriber network prefixes will normally be segmented across subscriber attachment points such as broadband or mobile gateways. This means that each SFC would have to be configured with the subscriber network prefixes whose traffic it is handling.

In a variation of the SFC configuration method describe above, the prefixes used in each direction can be such that they include all possible addresses at each side of the SFC. For example, in Figure 2, Network-A can be a prefix that includes all subscriber IP addresses and Network-B could be the default route, 0/0.

Using this technique, the same routes can be installed in all instances of an SFC that serve different groups of subscribers in different geographic locations.

The routes forwarding traffic into a SF instance and to the next SF instance are installed when an SFC is initially built, and each time a SF instance is connected into the SFC, but there is no requirement for VRFs to be reconfigured when traffic from different networks pass through the service chain, so long as their prefix is included in the prefixes in the VRFs along the SFC.

In this variation, it is assumed that no subscriber-originated traffic will enter the SFC destined for an IP address also in the subscriber network address range. This will not be a restriction in many cases.

2.9. Header Transforming Service Functions

If a service function performs an action that changes the source address in the packet header (e.g., NAT), the routes that were installed as described above may not support reverse flow traffic.

The solution to this is for the controller modify the routes in the reverse direction to direct traffic into instances of the transforming service function. The original routes with a source prefix (Network-A in Figure 2) are replaced with a route that has a prefix that includes all the possible addresses that the source address could be mapped to. In the case of network address translation, this would correspond to the NAT pool.

An example of this technique, combined with the prefix variation of the previous section, is provided in Appendix

3. Load Balancing Along a Service Function Chain

One of the key concepts driving NFV [NFVE2E]is the idea that each service function along an SFC can be separately scaled by changing the number of service function instances that implement it. This requires that load balancing be performed before entry into each service function. In this architecture, load balancing is performed in either or both of egress and ingress VRFs depending on the type of load balancing being performed, and if more than one service instance is connected to the same ingress VRF.

3.1. SF Instances Connected to Separate VRFs

If SF instances implementing a service in an SFC are each connected to separate VRFs(e.g. instances are connected to different routers

or are running on different hosts), load balancing is performed in the egress VRFs of the previous service, or in the VRF that is the entry to the SFC. The controller distributes BGP multi-path routes to the egress VRFs. The destination prefix of each route is the ultimate destination network. The next-hops in the ECMP set are BGP next-hops of the service instances attached to ingress VRFs of the next service in the SFC. The load balancing corresponds to BGP Multipath, which requires that the route distinguishers for each route are distinct in order to recognize that distinct paths should be used. Hence, each VRF in a distributed, SFC environment must have a unique route distinguisher.

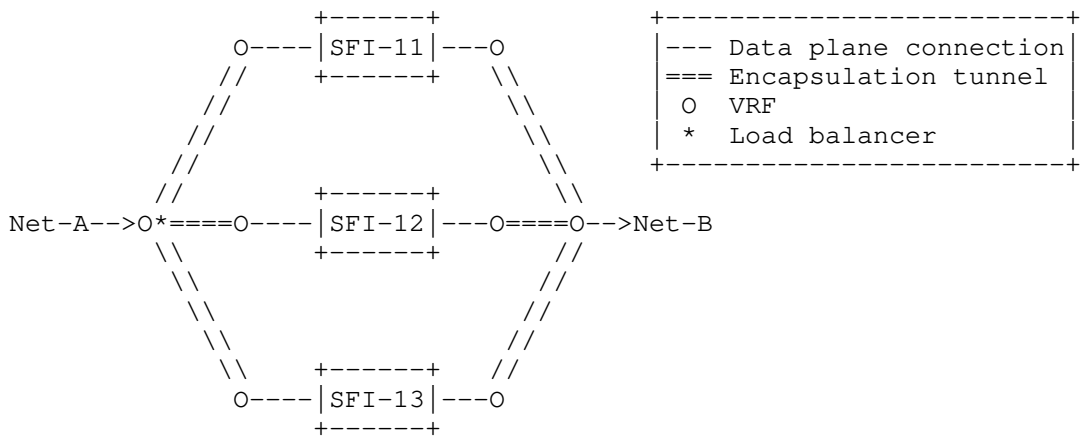


Figure 6 - Load Balancing across SF Instances Connected to Different VRFs

In the diagram, above, a service function is implemented in three service instances each connected to separate VRFs. Traffic from Network-A arrives at VRF at the start of the SFC, and is load balanced across the service instances using a set of ECMP routes with next hops being the addresses of the routing systems containing the ingress VRFs and with labels that identify the ingress interfaces of the service instances.

3.2. SF Instances Connected to the Same VRF

When SF instances implementing a service in an SFC are connected to the same ingress VRF, load balancing is performed in the ingress VRF across the service instances connected to it. The controller will install routes in the ingress VRF to the destination network with the interfaces connected to each service instance as next hops. The

ingress VRF will then use ECMP to load balance across the service instances.

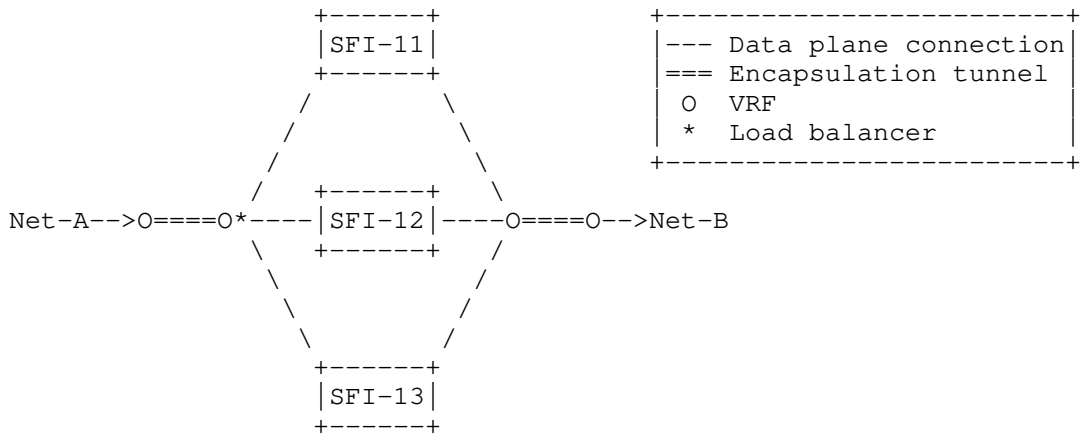


Figure 7 - Load Balancing across SF Instances Connected to the Same VRF

In the diagram, above, a service is implemented by three service instances that are connected to the same ingress and egress VRFs. The ingress VRF load balances across the ingress interfaces using ECMP, and the egress traffic is aggregated in the egress VRF.

3.3. Combination of Egress and Ingress VRF Load Balancing

In Figure 8, below, an example SFC is shown where load balancing is performed in both ingress and egress VRFs.

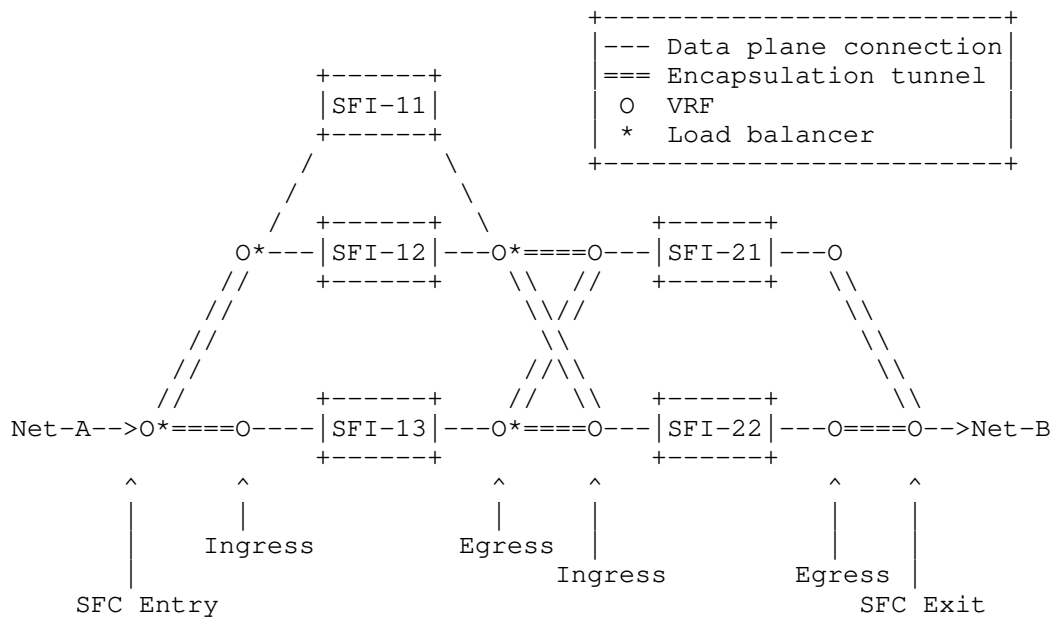


Figure 8 - Load Balancing across SF Instances

In Figure 8, above, an SFC is composed of two services implemented by three service instances and two service instances, respectively. The service instances SFI-11 and SFI-12 are connected to the same ingress and egress VRFs, and all the other service instances are connected to separate VRFs.

Traffic entering the SFC from Network-A is load balanced across the ingress VRFs of the first service function by the chain entry VRF, and then load balanced again across the ingress interfaces of SFI-11 and SFI-12 by the shared ingress VRF. Note that use of standard ECMP will lead to an uneven distribution of traffic between the three service instances (25% to SFI-11, 25% to SFI-12, and 50% to SFI-13). This issue can be mitigated through the use of BGP link bandwidth extended community [draft-ietf-idr-link-bandwidth].

After traffic passes through the first set of service instances, it is load balanced in each of the egress VRFs of the first set of service instances across the ingress VRFs of the next set of service instances.

3.4. Forward and Reverse Flow Load Balancing

This section discusses requirements in load balancing for forward and reverse paths when stateful service functions are deployed.

3.4.1. Issues with Equal Cost Multi-Path Routing

As discussed in the previous sections, load balancing in the forward SFC in the above example can automatically occur with standard BGP, if multiple equal cost routes to Network-B are installed into all the ingress VRFs, and each route directs traffic through a different service function instance in the next set. The multiple BGP routes in the routing table will translate to Equal Cost Multi-Path in the forwarding table. The hash used in the load balancing algorithm (per packet, per flow or per prefix) is implementation specific.

If a service function is stateful, it is required that forward flows and reverse flows always pass through the same service function instance. ECMP does not provide this capability, since the hash calculation will see different input data for the same flow in the forward and reverse directions (since the source and destination fields are reversed).

Additionally, if the number of SF instances changes, either increasing to expand capacity, or decreases (planned, or due to a SF instance failure), the hash table in ECMP is recalculated, and most flows will be directed to a different SF instance and user sessions will be disrupted.

There are a number of ways to satisfy the requirements of symmetric forward/reverse paths for flows and minimal disruption when SF instances are added to or removed from a set. Two techniques that can be employed are described in the following sections.

3.4.2. Modified ECMP with Consistent Hash

Symmetric forwarding into each side of an SF instance set can be achieved with a small modification to ECMP if the packet headers are preserved after passing through a SF instance set. In this case, each packet's 5-tuple data can be used in a hashing function, provided the source and destination IP address and port information are swapped in the reverse calculation and that the same or no hash salt is used for both directions. This method only requires that the list of available service function instances is consistently maintained in all the load balancers, rather than maintaining a distributed flow table.

In the SFC architecture described in this document, when SF instances are added or removed, the controller is required to configure (or remove) static routes to the SF instances. The controller could configure the load balancing function in VRFs that connect to each added (or removed) SF instance as part of the same

network transaction as route updates to ensure that the load balancer configuration is synchronized with the set of SF instances.

The effect of rehashing when SF instances are added or removed can be minimized, or even eliminated using variations of the technique of consistent hashing [consistent-hash]. Details are outside the scope of this document.

3.4.3. ECMP with Flow Table

A second refinement that can ensure forward/reverse flow consistency, and also provides stability when the number of SF instances changes ("flow-stickiness"), is the use of dynamically configured IP flow tables in the VRFs. In this technique, flow tables are used to ensure that existing flows are unaffected if the number of ECMP routes changes, and that forward and reverse traffic passes through the same SF instance in each set of SF instances implementing a service function.

The flow tables are set up as follows:

1. User traffic with a new 5-tuple enters an egress VRF from a connected SF instance.

2. The VRF calculates the ECMP hash across available routes (i.e., ECMP group) to the ingress interfaces of the SF instances in the next SF instance set.

3. The VRF creates a new flow entry for the 5-tuple traffic with the next-hop being the chosen downstream ECMP group member (determined in the step 2. above) . All subsequent packets for the same flow will be forwarded using flow lookup and, hence, will use the same next-hop.

4. The encapsulated packet arrives in the routing system that hosts the ingress VRF for the selected SF instance.

5. The ingress VRF of the next service instance determines if the packet came from a routing system that is in an ECMP group in the reverse direction(i.e., from this ingress VRF back to the previous set of SF instances).

6. If an ECMP group is found, the ingress VRF creates a reverse flow entry for the 5-tuple with next-hop of the tunnel on which traffic arrived.

7. The packet is sent into the SF instance connected to the ingress VRF.

The above method ensures that forward and reverse flows pass through the same SF instances, and that if the number of ECMP routes changes when SF instances are added or removed, all existing flows will continue to flow through the same SF instances, but new flows will use the new ECMP hash. The only flows affected will be those that were passing through an SF instance that was removed, and those will be spread among the remaining SF instances using the updated ECMP hash.

4. Steering into SFCs Using a Classifier

In many applications of SFCs, a classifier will be used to direct traffic into SFCs. The classifier inspects the first or first few packets in a flow to determine which SFC the flow should be sent into. The decision criteria can include the IP 5-tuple of the header, and/or analysis of the payload of packets using deep packet inspection. Integration with a subscriber management system such as PCRF or AAA will usually be required in order to identify which SFC to send traffic to based on subscriber policy.

An example logical architecture is shown in Figure 9, below where a classifier is external to a physical router.

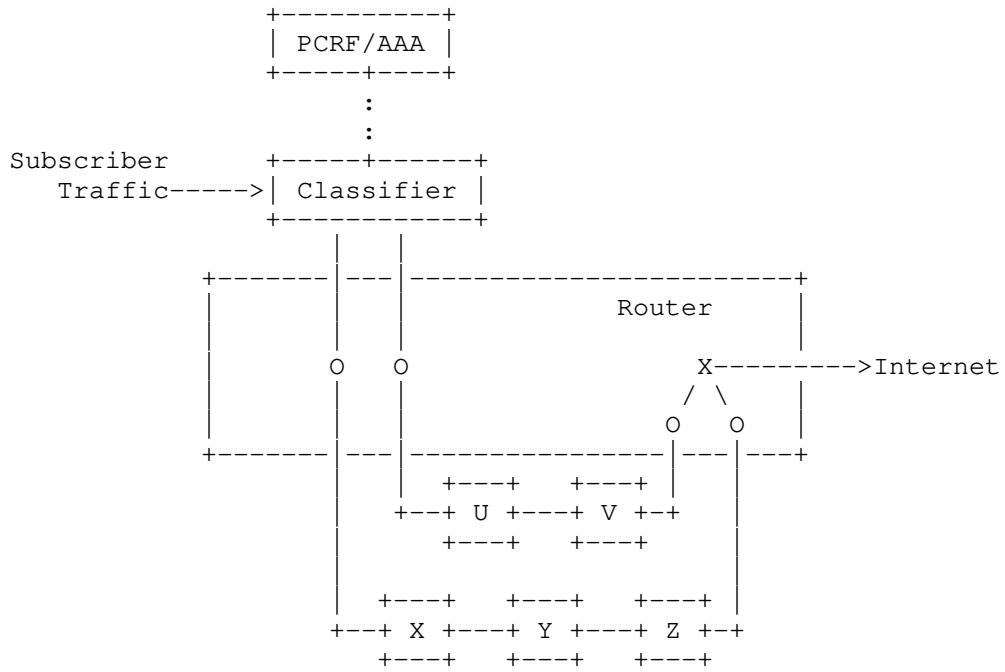


Figure 9- Subscriber/Application-Aware Steering with a Classifier

In the diagram, the classifier receives subscriber traffic and sends the traffic out of one of two logical interfaces, depending on classification criteria. The logical interfaces of the classifier are connected to VRFs in a router that are entries to two SFCs (shown as O in the diagram).

In this scenario, the exit VRF for each SFC does not peer with a gateway or proxy node in the destination network and packets are forwarded using IP lookup in the main routing table or in a VRF that the exit traffic from the SFCs is directed into (shown as X in the diagram).

An alternative would be where the classifier is itself a distributed, virtualized service function, but with multiple egress interfaces. In that case, each virtual classifier instance could be attached to a set of VRFs that connect to different SFCs. Each chain entry VRF would load balance across the first SF instance set in its SFC. The reverse flow table mechanism described in Section 4.3 could be employed to ensure that flows return to the originating classifier instance which may maintain subscriber context and perform charging and accounting.

5. Controller Federation

It is likely that SFCs will be managed as a separate administrative domain from the networks that they receive traffic from, and send traffic to. If the connected networks use BGP for route distribution, the controller in the SFC domain can join the network domains by creating BGP peering sessions with routing systems or route reflectors in the network domains.

When SFCs are distributed geographically, or in very large-scale environments, there may be multiple SFC controllers present. If there is a requirement for SFCs to span controller domains there may be a requirement to exchange information between controllers. Again, a BGP session between controllers can be used to exchange route information and allow such domain spanning SFCs to be created.

6. Summary and Conclusion

The architecture for service function chains described in this document uses virtual networks implemented as overlays in order to create service function chains. The virtual networks use standards-based encapsulation tunneling, such as MPLS over GRE or VXLAN, to transport packets into an SFC and between service function instances without routing in the user address space. The controller contains a topological model of the SFC that includes the connections from SF instances to routing systems and the required connectivity between

service functions. VRFs with common route targets are used to define the virtual networks that connect sets of SF instances to form the SFC. The SF instances are linked to VRFs by installing static routes that direct traffic through the SF instances. BGP route-reflection is used to distribute these routes and form service function chains.

The architecture can be implemented on today's routers without modification to existing signaling protocols, and without modification to the operation of the routers themselves. There is no requirement for a service chain header to be added to packets, and no requirement for any service chain topology, other than next hops, to be sent to routing systems.

In environments with physical routers, the controller may operate in tandem with existing BGP route reflectors, and would contain the SFC topology model, and the ability to install the local static interface routes to SF instances. In a virtualized environment, the controller can emulate route refection internally and simply install required routes directly without advertisements occurring.

7. Security Considerations

The security considerations for SFCs are broadly similar to those concerning the data, control and management planes of any device placed in a network. Details are out of scope for this document.

8. IANA Considerations

There are no IANA considerations.

9. References

9.1. Normative References

None

9.2. Informative References

[NFVE2E] "Network Functions Virtualisation: End to End Architecture, http://docbox.etsi.org/ISG/NFV/70-DRAFT/0010/NFV-0010v016.zip".

[RFC2328] J. Moy, "OSPF Version 2", RFC 2328, April, 1998.

[draft-quinn-sfc-arch] Quinn, P. and A. Beliveau, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-04 (work in progress), January 2014.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

[draft-mahalingam-vxlan] M. Mahalingam, et al. "VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks.", draft-mahalingam-dutt-dcops-vxlan-08, February 3, 2014.

[draft-bouadair-sfc-arch] Boucadair, M., and Jacquenet, C., "Service Function Chaining: Framework & Architecture", draft-boucadair-sfc-framework-02, February 2014.

[draft-quinn-sfc-arch]
P. Quinn, at al, "Service Function Chaining (SFC) Architecture", draft-quinn-sfc-arch-04, January 28, 2014.

[draft-ietf-l3vpn-end-system]
P. Marques et al., "BGP-signaled end-system IP/VPNs", draft-ietf-l3vpn-end-system, October 21, 2013.

[draft-quinn-sfc-nsh]
Quinn, P., et al, "Network Service Header", draft-quinn-sfc-nsh-01, February 2014.

[draft-niu-sfc-mechanism]
Niu, L., Li, H., and Jiang, Y., "A Service Function Chaining Header and its Mechanism", draft-niu-sfc-mechanism-00, January 2014.

[draft-rijsman-sfc-metadata-considerations]
B. Rijsman, et al. "Metadata Considerations", draft-rijsman-sfc-metadata-considerations-00, February 12, 2014

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

[RFC4023] Worster, T., Rekhter, Y., and E. Rosen, "Encapsulating MPLS in IP or Generic Routing Encapsulation (GRE)", RFC 4023, March 2005.

[draft-ietf-mpls-in-udp]
Building, K., Sheth, N., Yong, L., Pignataro, C., and F. Yongbing, "Encapsulating MPLS in UDP", draft-ietf-mpls-in-udp-03 (work in progress), September 2013.

[RFC3031] Rosen. E, Viswanathan A, Callon R, "Multiprotocol Label Switching Architecture", January 2001.

[draft-ietf-i2rs-architecture]
Atlas, A., Halpern, J., Hares, S., Ward, D., and T Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture, work in progress.

[consistent-hash]
Karger, D.; Lehman, E.; Leighton, T.; Panigrahy, R.; Levine, M.; Lewin, D. (1997). "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web". Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing. ACM Press New York, NY, USA. pp. 654-663.

[draft-ietf-idr-link-bandwidth]
P. Mohapatra, R. Fernando, "BGP Link Bandwidth Extended Community", draft-ietf-idr-link-bandwidth, work in progress.

10. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

The authors would like to thank Nischal Seth and Galina Pildush of Juniper Networks, and Nabil Bitar of Verizon for their review and comments.

Appendix A.SFC Implementation in a Worked Example

A simple service function chain with just one instance of one service function in it will be used in order to describe, in detail, the actions of the control and forwarding planes in this architecture. The example will then be generalized to show how more complex SFCs with multiple service functions, each with multiple SF instances, can be implemented.

It is assumed, in the example, that the routing systems in the network fully support BGP route signaling (receiving advertisements and sending updates), and that Netconf can be used for routing system configuration. Netconf (or other configuration protocol) is required because the current operation of BGP does not support installation of local routes using an interface identifier as a next hop.

As described in Section 7.3, a controller in a virtualized environment can calculate required routes without receiving route advertisements, and the resulting installed routes will be the same as described below.

A.1. Desciption of Example SFC Topology

The example SFC is shown in Figure 10, below. Two service functions are connected serially, and each is implemented in one service instance. Traffic flowing between Network-A and Network-B should pass through SFI-1 and SFI-2.

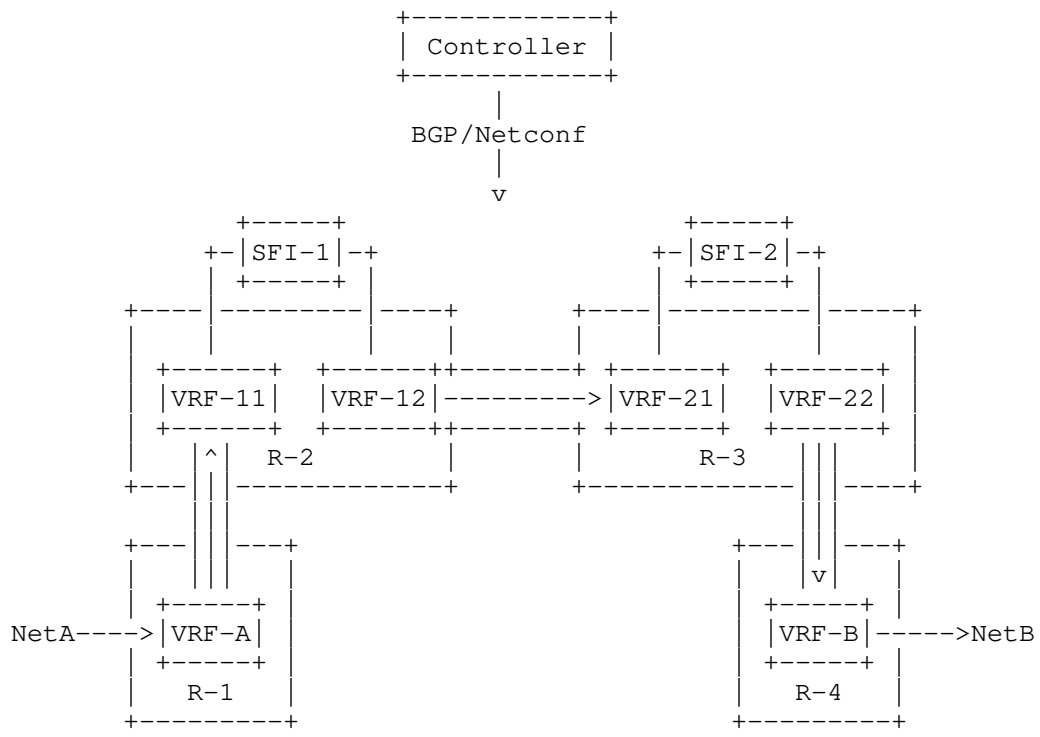


Figure 10- Service Chain with Two Service Functions

The SFC in the diagram is one half of an SFC set that allows traffic in both directions to pass through the service functions. In the example below, the routes for both directions will be described.

A.1.1. Connectivity in the Example

Routing systems R-1, R-2, R-3 and R-4 are in the same autonomous system and have IP connectivity with each other. The routing systems may be directly connected, or be connected via an Ethernet or IP transport network. The routing systems may be physical or virtual.

VRFs VRF-A, VRF-11, VRF-12, VRF-21, VRF-22 and VRF-B are created (using, for instance, Netconf) on routing systems R-1, R-2, R-3 and R-4.

Interface IF-NetA in system R-1 is present in VRF-A, and connects to Network-A. Similarly, IF-NetB in system R-4 is present in VRF-B, and connects to Network-B.

Service function instance SFI-1 is connected via interfaces on R-2 named IF-11 and IF-12, and which are present in VRF-11 and VRF-12, respectively.

Service function instance SFI-2 is similarly connected to VRF-21 and VRF-22 in routing system R-3.

VRF-A is the entry point into the service function chain and VRF-B is the exit point for traffic flowing from Network-A to Network-B.

The service function instances, SFI-1 and SFI-2 may be directly connected to routing systems R-2 and R-3, or may be connected via a L2 switching network.

In the example, encapsulation tunneling takes place between VRF-A and VRF-11, between VRF-12 and VRF-21, and between VRF-22 and VRF-B.

A.2. Forwarding Control

The forwarding of a packet from Network-A to a destination in Network-B through the service function chain is achieved by the installation by the controller of local routes in the ingress routing systems followed by route reflection to connected egress routing systems.

The combination of the specially installed routes together with conventional route reflection enables the required flow through the virtual networks and through the service function instances.

A.2.1. Initial Local Routes

The initial state is that each router contains routes to the other routers in its main IP routing table, and routes to Network-A and Network-B will be in the tables of the connected VRFs, VRF-A and VRF-B. The routes to Network-A and Network-B may be statically configured, or signaled via a routing protocol from systems in these networks or from gateways to these networks. The routes between the routers in the SFC will have been signaled by some interior gateway protocol, such as OSPF [RFC2328].

The following table shows the import and export policies that are configured when the SF instances are connected.

Routing System	VRF	Import Target	Export Target
R-1	VRF-A	RTA	RTA
R-2	VRF-11	RTA	RTA
R-2	VRF-12	RT12	RT12
R-3	VRF-21	RT12	RT12
R-3	VRF-22	RTB	RTB
R-4	VRF-B	RTB	RTB

These import and export policies allow route exchange between VRF-A and VRF-11 and between VRF-22 and VRF-B.

The following table shows the relevant local routes in the routing systems R-1, R-2, R-3 and R-4.

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Network-A	IF-NetA
R-1	Global	R-2	IF-R-1-2
R-2	Global	R-1	IF-R-2-1
R-2	Global	R-3	IF-R-2-3
R-3	Global	R-2	IF-R-3-2
R-3	Global	R-4	IF-R-3-4
R-4	Global	R-3	IF-R-4-3
R-4	VRF-B	Network-B	IF-NetB

The term "Global" in the above table means that route is installed in the main routing table of the routing system.

A naming convention is used for the interfaces that connect routers, so, for instance, interface IF-R-1-2 is the interface on R-1 that connects to R-2.

A.2.2. Route Advertisements to Build Virtual Networks

In the standard way, each routing system sends advertisements for each of its local routes to the controller (acting as a route

reflector here). The L3 VPN advertisements for the network in Figure 1 are shown in the following table.

Routing System	Advertised Prefix	Label	Route Distinguisher	Route Target
R-1 R-4	Network-A Network-B	Lbl-NetA Lbl-NetB	RDA RDB	RTA RTB

The labels are locally significant on each routing system, and are used to identify the interface that incoming labeled packets will be sent out on. [RFC4364].

Using standard route reflection, route updates are sent to each routing system, and routes are installed in VRFs with matching import route targets.

The following additional routes would be installed in the VRFs on R-2 and R-3:

Routing System	VRF	Prefix	BGP Next Hop
R-2	VRF-11	Network-A	push Lbl-NetA, encap GRE, R-1
R-3	VRF-22	Network-B	push Lbl-NetB, encap GRE, R-4

The first new route entry states that for packets with the prefix of Network-A received in VRF-11 in routing system R-2, the next hop action is to push the advertised label (Lbl-NetA), encapsulate with GRE, and send the GRE packet to router R-1. The destination R-1 will be resolved to interface IF-R-2-1 in the forwarding table. The second entry is similar, but for Network-B packets arriving in VRF-22.

At this point, virtual networks corresponding to route targets RTA and RTB exist, and VRF VRF-11 can reach Network-A, and VRF-12 can reach Network-B, but Network-A is not reachable from Network-B, or vice versa.

A.3. Enabling Forwarding into SF Instances by the Controller

In this architecture, the controller has a topological model of each SFC. The model contains the service function instances that implement the SFC together with the desired connectivity to VRFs via

specific interfaces in the routing systems. The controller uses that information to determine which local routes to install, in order to have traffic from networks on one side of an SFC reach networks on the other side by passing through the SFC.

The controller uses Netconf to install the routes, since the next hop is an interface name, not an IP address.

A.4. Local Route Installation

A local route is installed in each VRF that connects to a SF instance. The local routes have the prefix of a connected network and a next hop that identifies a local interface that is connected to a SF instance.

In the example SFC, the controller sends the following local routes to routing system R-2:

Routing System	Prefix	Direct Next Hop	Route Dist	Route Target
R-2	Network-B	Local IF-11	RD11	RTA
R-2	Network-A	Local IF-12	RD12	RT12
R-3	Network-B	Local IF-21	RD21	RT12
R-3	Network-A	Local IF-22	RD22	RTB

Today, BGP does not support installation of routes containing an interface identifier as a next hop, and these would need to be installed via Netconf or other means, such as I2RS or XMPP [draft-ietf-i2rs-architecture, draft-ietf-l3vpn-end-system].

The following new routes will exist on R-2 and R-3:

Routing System	VRF	Prefix	Next Hop
R-2	VRF-11	Network-B	Local IF-11
R-2	VRF-12	Network-A	Local IF-12
R-3	VRF-21	Network-B	Local IF-21
R-3	VRF-22	Network-A	Local IF-22

Routing systems R-2 and R-3 will advertise new routes back to the controller since the interfaces are MPLS enabled and present in the VRFs connected to SF instances:

Routing System	Advertised Prefix	Label	Route Dist	Route Target
R-2	Network-B	Lbl-IF-11	RD11	RTA
R-2	Network-A	Lbl-IF-12	RD12	RT12
R-3	Network-B	Lbl-IF-21	RD21	RT12
R-3	Network-A	Lbl-IF-22	RD22	RTB

Having received these advertisements from R-2 and R-3, the controller, acting as route reflector, will send the following route updates:

Prefix	BGP Next Hop	Label	Route Dist	Route Target
Network-B	R-2	Lbl-IF-11	RD11	RTA
Network-A	R-2	Lbl-IF-12	RD12	RT12
Network-B	R-3	Lbl-IF-21	RD21	RT12
Network-A	R-3	Lbl-IF-22	RD22	RTB

These would be resolved in the various routing systems as follows:

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Network-B	push Lbl-IF-11, encap GRE, R-2
R-2	VRF-12	Network-B	push Lbl-IF-21, encap GRE, R-3
R-3	VRF-21	Network-A	push Lbl-IF-12, encap GRE, R-2
R-4	VRF-B	Network-A	push Lbl-IF-22, encap GRE, R-3

The routes installed in each router will be as follows:

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Network-A	IF-NetA
R-1	VRF-A	Network-B	push Lbl-IF-11, encap GRE, R-2
R-1	Global	R-2	IF-R-1-2
R-2	VRF-11	Network-B	local IF-12
R-2	VRF-11	Network-A	push Lbl-NetA, encap GRE, R-1
R-2	Global	R-1	IF-R-1-2
R-2	VRF-12	Network-A	local IF-21
R-2	VRF-12	Network-B	push Lbl-IF-21, encap GRE, R-3
R-2	Global	R-3	IF-R-2-3
R-3	VRF-21	Network-B	local IF-12
R-3	VRF-21	Network-A	push Lbl-IF-12, encap GRE, R-2
R-3	Global	R-2	IF-R-3-2
R-3	VRF-22	Network-A	local IF-21
R-3	VRF-22	Network-B	push Lbl-NetB, encap GRE, R-4
R-3	Global	R-4	IF-R-3-4
R-4	VRF-B	Network-A	push Lbl-IF-22, encap GRE, R-3
R-4	Global	R-3	IF-R-4-3
R-4	VRF-B	Network-B	IF-NetB

The MPLS tables will be as follows:

Routing System	Label	Action	Direct Next Hop
R-1	Lbl-NetA	pop	IF-NetA
R-2	Lbl-IF-11	pop	IF-11
R-2	Lbl-IF-12	pop	IF-12
R-3	Lbl-IF-21	pop	IF-21
R-3	Lbl-IF-22	pop	IF-22
R-4	Lbl-NetB	pop	IF-NetB

A connection is established between Network-A and Network-B with packets flowing through SF instances SFI-1 and SFI-2 as a result of adding the local routes that point to the interfaces in the routing systems that the SF instance is connected to is that.

A.5. Detailed Packet Flow

This section describes the details of how a packet is forwarded from Network-A to Network-B through the example SFC. The table, below, shows the routes for one direction only.

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Network-B	push Lbl-IF-11, encap GRE, R-2
R-1	Global	R-2	IF-R-1-2
R-2	VRF-11	Network-B	local IF-12
R-2	VRF-12	Network-B	push Lbl-IF-21, encap GRE, R-3
R-2	Global	R-3	IF-R-2-3
R-3	VRF-21	Network-B	local IF-21
R-3	VRF-22	Network-B	push Lbl-NetB, encap GRE, R-4
R-3	Global	R-4	IF-R-3-4
R-4	VRF-B	Network-B	IF-NetB

The MPLS table entries for this direction are:

Routing System	Label	Action	Direct Next Hop
R-2	Lbl-IF-11	pop	IF-11
R-3	Lbl-IF-21	pop	IF-21
R-4	Lbl-NetB	pop	IF-NetB

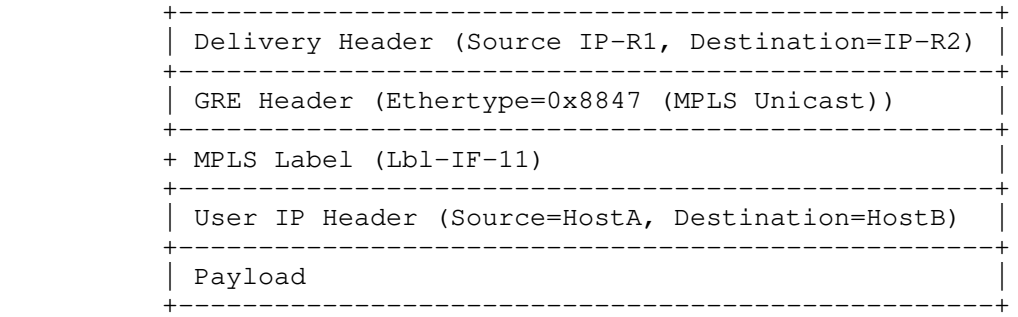
The detailed packet forwarding, labeling and encapsulation for a packet from a host HostA in Network-A to HostB in Network-B would be as follows:

1. System R-1 is advertising a route for the prefix corresponding to Network-B into Network-A, or a gateway in Network-A has a route configured such that packets originating in Network-A with destination in Network-B will be sent to R-1.

2. An IP packet arrives at interface IF-NetA on system R-1 with a header containing HostB as the destination IP address and HostA as the source IP address.

3. Interface IF-NetA on R-1 is configured in VRF VRF-A, so the packets arrive in VRF-A, whose forwarding table directs it to push a label Lbl-IF-11 onto the packet and use GRE encapsulation to send the packet to R-2.

4. A GRE header is added and the packet now has the following structure:



where IP-R1 and IP-R2 are the addresses used for the connection between R-1 and R-2. These could be loopback addresses, interface addresses or server addresses, depending on the implementation.

5. The encapsulated packet arrives at R-2. The outer header is stripped off together with the GRE header. The label is looked up in the MPLS table where the action is to pop the label and to send to IF-11.

6. The packet exits IF-11 on R-2 and arrives at the ingress interface of the SF instance SFI-1. The packet passes through SFI-1, where the internal logic of SFI-1 processes the packet information. Since, in this case, SFI-1 is a transparent service function, the packet emerges with the header unchanged (assuming the logic did not determine that the packet should be dropped).

7. The packet is sent from the egress interface of SFI-1 to interface IF-12 on system R-2, which is configured in VRF-12.

8. The forwarding table in VRF-12 directs that a label Lbl-IF-21 be pushed onto the packet header and GRE encapsulation is used to send the packet to R-3.

9. A GRE encapsulation header is added to the original IP header with source IP of IP-R2 and destination address IP-R3. The encapsulated packet is sent to R-3.

10. The packet arrives at R-3, where the GRE header is removed, and the label Lbl-IF-21 is looked up in the MPLS table where the action is to pop the label and send out of interface IF-21.

11. The packet exits IF-21 on R-3 and arrives at the ingress interface of the SF instance SFI-2. The packet passes through SFI-2, where the internal logic of SFI processes the packet information.

12. The packet is sent from the egress interface of SFI-2 to interface IF-22 on system R-3, which is configured in VRF-22.

13. The forwarding table in VRF-22 directs that a label Lbl-NetB be pushed onto the packet header and GRE encapsulation is used to send the packet to R-4.

14. A GRE encapsulation header is added to the original IP header with source IP of IP-R3 and destination address IP-R4. The encapsulated packet is sent to R-4.

15. The packet arrives at R-4, where the GRE header is removed, and the label LBL-NetB is looked up in the MPLS table where the action is to pop the label and send to IF-NetB.

16. The packet is then sent with the original IP header out of interface IF-NetB on R-4 towards the destination HostB in Network-B.

Traffic flowing the reverse direction from HostB to HostA will follow the same path, but in reverse.

A.6. Extended SFCs

Following the pattern described above, when an SFC has more than two service functions, a different route target is used for the VRFs that connect each pair of SF instances. Routes pointing to the connected SF instance interface are installed in each VRF, and route refection causes the installation of corresponding routes for traffic entering VRFs in the same virtual network from the egress of a SF instance. The routes installed in R-2 and R-3 are essentially replicated for each connection between SF instances to form the SFC.

A.7. SFCs Using Routes with Expanded Prefixes

In a variation to the method described above, the routes in each direction can be set to prefixes that include all possible traffic.

In the example SFC, Network-A can be a prefix that includes all subscriber IP addresses and Network-B could set to the default route, 0/0.

With this variation, routes forwarding traffic into a SF instance and to the next SF instance are installed each time a SF instance is connected, but there is no requirement for VRFs to be reconfigured when traffic from new networks pass through the service chain.

If a classifier is used to direct traffic into SFCs, the same SFC configuration can be used in all instances of an SFC in a network, thus simplifying testing and deployment.

A.8. SFCs with Packet Transforming Service Functions

If a service function performs an action that changes the source address in the packet header, the routes that were installed as described above may not support reverse flow traffic.

The solution to this is for the controller modify the routes in the reverse direction that direct traffic into instances of the transforming service function. The original routes with a source prefix (Network-A in the example) are replaced with a route that has a prefix that includes all the possible addresses that the source address could be mapped to. In the case of network address translation, this would correspond to the NAT pool.

The following table shows the routes that would be installed in the example SFC network if the second service function transformed the packet headers, and technique described in Section was employed. The subscriber address pool is Sub-Pool, and NAT-Pool is the public address space that the second service function uses.

Routing System	VRF	Prefix	Next Hop
R-1	VRF-A	Sub-Pool	IF-NetA
R-1	VRF-A	0/0	push Lbl-IF-11, encap GRE, R-2
R-1	Global	R-2	IF-R-1-2
R-2	VRF-11	0/0	local IF-12
R-2	VRF-11	Sub-Pool	push Lbl-NetA, encap GRE, R-1
R-2	Global	R-1	IF-R-1-2
R-2	VRF-12	Sub-Pool	local IF-21
R-2	VRF-12	0/0	push Lbl-IF-21, encap GRE, R-3
R-2	Global	R-3	IF-R-2-3
R-3	VRF-21	0/0	local IF-12
R-3	VRF-21	Sub-Pool	push Lbl-IF-12, encap GRE, R-2
R-3	Global	R-2	IF-R-3-2
R-3	VRF-22	NAT-Pool	local IF-21
R-3	VRF-22	0/0	push Lbl-NetB, encap GRE, R-4
R-3	Global	R-4	IF-R-3-4
R-4	VRF-B	NAT-Pool	push Lbl-IF-22, encap GRE, R-3
R-4	Global	R-3	IF-R-4-3
R-4	VRF-B	0/0	IF-NetB

Authors' Addresses

Stuart Mackie
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

Email: wsmackie@juniper.net

Bruno Rijsman
Juniper Networks
1133 Innovation Way
Sunnyvale, CA 94089
USA

Email: brijsman@juniper.net

Maria Napierala
AT&T Labs
200 Laurel Avenue
Middletown, NJ 07748

Email: mnapierala@att.com

Diego Daino
Telecom Italia
Via Guglielmo Reiss Romoli
274 - 10148 Turin
Italy

Email: diego.daino@telecomitalia.it

Diego R. Lopez
Telefonica I+D
Don Ramon de la Cruz, 82
Madrid 28006
Spain

Email: diego@tid.es

Daniel Bernier
Bell Canada
1 Carrefour Alexander-Graham Bell
Building A-7,Verdun
Quebec, H3E 3B3
Canada

Email: daniel.bernier@bell.ca

Walter Haeffner
Vodafone D2 GmbH
Ferdinand-Braun-Platz 1
Dusseldorf 40549
DE

Email: walter.haeffner@vodafone.com

L2VPN Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 8, 2015

S. Mohanty
K. Patel
A. Sajassi
Cisco Systems, Inc.
J. Drake
Juniper Networks, Inc.

March 7, 2015

A new Designated Forwarder Election for the EVPN
draft-mohanty-l2vpn-evpn-df-election-01

Abstract

This document describes an improved EVPN Designated Forwarder Election (DF) algorithm which can be used to enhance operational experience in terms of convergence speed and robustness over a WAN deploying EVPN

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 8, 2015.

Copyright Notice

Table of Contents

1. Introduction

Ethernet MPLS VPN (EVPN) [RFC7432]is an emerging technology that is gaining prominence in Internet Service Provider IP/MPLS networks. In EVPN, mac addresses are disseminated as routes across the geographical area via the Border Gateway Protocol, BGP [RFC4271] using the familiar L3VPN model [RFC4364]. An EVPN instance that spans across PEs is defined as an EVI. Constrained Route Distribution [RFC4684] can be used in conjunction to selectively advertise the routes to where they are needed. One of the major advantages of EVPN over VPLS [RFC4761],[RFC6624] is that it provides a solution for minimizing flooding of unknown traffic and also provides all Active mode of operation so that the traffic can truly be multi-homed. In technologies such as EVPN or VPLS, managing Broadcast, Unknown Unicast and multicast traffic (BUM) is a key requirement. In the case where the customer edge (CE) router is multi-homed to one or more Provider Edge (PE) Routers, it is necessary that one and only one of the PE routers should forward BUM traffic into the core or towards the CE as and when appropriate.

Specifically, quoting Section 8.5, [RFC7432], Consider a CE that is a host or a router that is multi-homed directly to more than one PE in an EVPN instance on a given Ethernet segment. One or more Ethernet Tags may be configured on the Ethernet segment. In this scenario only one of the PEs, referred to as the Designated Forwarder (DF), is responsible for certain actions:

a. Sending multicast and broadcast traffic, on a given Ethernet Tag on a particular Ethernet segment, to the CE.

b. Flooding unknown unicast traffic (i.e. traffic for which an PE does not know the destination MAC address), on a given Ethernet Tag on a particular Ethernet segment to the CE, if the environment requires flooding of unknown unicast traffic.

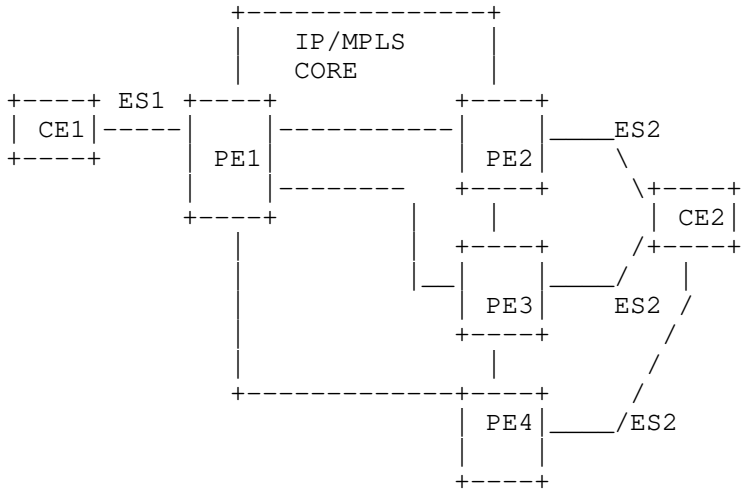


Figure 1 Multi-homing Network of E-VPN

Figure 1

Figure 1 illustrates a case where there are two Ethernet Segments, ES1 and ES2. PE1 is attached to CE1 via Ethernet Segment ES1 whereas PE2, PE3 and PE4 are attached to CE2 via ES2 i.e. PE2, PE3 and PE4 form a redundancy group. Since CE2 is multi-homed to different PEs on the same Ethernet Segment, it is necessary for PE2, PE3 and PE4 to agree on a DF to satisfy the above mentioned requirements.

Layer2 devices are particularly susceptible to forwarding loops because of the broadcast nature of the Ethernet traffic. Therefore it is very important that in case of multi-homing, only one of the links be used to direct traffic to/from the core.

One of the pre-requisites for this support is that participating PEs must agree amongst themselves as to who would act as the Designated Forwarder. This needs to be achieved through a distributed algorithm

in which each participating PE independently and unambiguously selects one of the participating PEs as the DF, and the result should be unanimously in agreement.

The DF election algorithm as described in the base EVPN draft has some undesirable properties and in some cases can be somewhat disruptive and unfair. This document describes those issues and proposes a mechanism for dealing with those issues. These mechanisms do involve changes to the DF Election algorithm , but do not require any protocol changes to the EVPN Route exchange and have minimal changes to their content per se.

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 [RFC2119].

2. The modulus based DF Election Algorithm

The default procedure for DF election at the granularity of (ESI,EVI) is referred to as "service carving". With service carving, it is possible to elect multiple DFs per Ethernet Segment (one per EVI) in order to perform load-balancing of multi-destination traffic destined to a given Segment. The objective is that the load-balancing procedures should carve up the EVI space among the redundant PE nodes evenly, in such a way that every PE is the DF for a disjoint set of EVIs.

The existing DF algorithm as described in the EVPN RFC(Section 8.5 [RFC7432]) is based on a modulus operation. The PEs to which the ES (for which DF election is to be carried out per vlan) is multi-homed form an ordered (ordinal) list in ascending order of the PE ip address values. Say, there are N PEs, P0, P1, ... PN-1 ranked as per increasing IP addresses in the ordinal list; then for each vlan with ethernet tag v, configured on the ethernet segment ES1, PEx is the DF for vlan v on ES ES1 when x equals (v mod N). In the case when the vlan density is high meaning there are significant number of vlans and the vlan-id or ethernet-tag is uniformly distributed, the thinking is that the DF election will be spread across the PEs hosting that ethernet segment and good service carving can be achieved.

3. Problems with the modulus based DF Election Algorithm

There are three fundamental problems with the current DF Election.

First, the algorithm will not perform well when the ethernet tag follows a non-uniform distribution, for instance when the ethernet tags are all even or all odd. In such a case let us assume that the ES is multi-homed to two PEs; all the vlans will only pick one of the PEs as the DF. This is very sub-optimal. It defeats the purpose of service carving as the DFs are not really evenly spread across. In this particular case, in fact one of the PEs does not get elected all as the DF, so it does not participate in the DF responsibilities at all. Consider another example where referring to Figure 1, lets assume that PE2, PE3, PE4 are in ascending order of the IP address; and each vlan configured on ES2 is associated with an Ethernet Tag of of the form (3x+1), where x is an integer. This will result in PE3 always be selected as the DF.

Even in the case when the ethernet tag distribution is uniform the instance of a PE being up or down results in re-computation ((v mod N-1) or (v mod N+1) as is the case); The resulting modulus value need not be uniformly distributed but subject to the primality of N-1 or N+1 as may be the case.

The third problem is one of disruption. Consider a case when the same Ethernet Segment is multi homed to a set of PEs. When the ES is down in one of the PEs, say PE1, or PE1 itself reboots, or the BGP process goes down or the connectivity between PE1 and an RR goes down, the effective number of PEs in the system now becomes N-1 and DFs are computed for all the vlans that are configured on that ethernet segment. In general, if the DF for a vlan v happens not to be PE1, but some other PE, say PE2, it is likely that some other PE will become the new DF. This is not desirable. Similarly when a new PE hosts the same Ethernet segment, the mapping again changes because of the mod operation. This results in needless churn. Again referring to Figure 1, say v1, v2 and v3 are vlans configured on ES2 with associated ethernet tags of value 999, 1000 and 10001 respectively. So PE1, PE2 and PE3 are also the DFs for v1, v2 and v3 respectively. Now when PE3 goes down, PE2 will become the DF for v1 and PE1 will become the DF for v2.

One point to note is that the current DF election algorithm assumes that all the PEs who are multi-homed to the same Ethernet Segment and interested in the DF Election by exchanging EVPN routes have a V4 peering with each other or via a Route Reflector. This need not be the case as there can be a v6 peering and supporting the EVPN address-family.

Mathematically, a conventional hash function maps a key k to a number i representing one of m hash buckets through a function h(k) i.e. i=h(k). In the EVPN case, h is simply a modulo-m hash function viz. h(v) = v mod N, where N is the number of PEs that are multi-homed to

the Ethernet Segment in discussion. It is well-known that for good hash distribution using the modulus operation, the modulus N should be a prime-number not too close to a power of 2 [CLRS2009]. When the effective number of PEs changes from N to N-1 (or vice versa); all the objects (vlan v) will be remapped except those for which v mod N and v mod (N-1) refer to the same PE in the previous and subsequent ordinal rankings respectively.

From a forwarding perspective, this is a churn, as it results in programming the CE and PE side ports as blocking or non-blocking at potentially all PEs when the DF changes either because (i) a new PE is added or (ii) another one goes down or loses connectivity or else cannot take part in the DF election process for whatever reason. This draft addresses this problem and furnishes a solution to this undesirable behavior.

4. Highest Random Weight

Highest Random Weight (HRW) as defined in [HRW1999] is originally proposed in the context of Internet Caching and proxy Server load balancing. Given an object name and a set of servers, HRW maps a request to a server using the object-name (object-id) and server-name (server-id) rather than the state of the server states. HRW forms a hash out of the server-id and the object-id and forms an ordered list of the servers for the particular object-id. The server for which the hash value is highest, serves as the primary responsible for that particular object, and the server with the next highest value in that hash serves as the backup server. HRW always maps a given object object name to the same server within a given cluster; consequently it can be used at client sites to achieve global consensus on object-server mappings. When that server goes down, the backup server becomes the responsible designate.

Choosing an appropriate hash function that is statistically oblivious to the key distribution and imparts a good uniform distribution of the hash output is an important aspect of the algorithm,. Fortunately many such hash functions exist. [HRW1999] provides pseudorandom functions based on Unix utilities rand and srand and easily constructed XOR functions that perform considerably well. This imparts very good properties in the load balancing context. Also each server independently and unambiguously arrives at the primary server selection. HRW already finds use in multicast and ECMP [RFC2991],[RFC2992].

In the existing DF algorithm Section 2, whenever a new PE comes up or an existing PE goes down, there is a significant interval before the change is noticed by all peer PEs as it has to be conveyed by the BGP update message involving the type-4 route. There is a timer to batch

all the messages before triggering the service carving procedures. When the timer expires, each PE will build the ordered list and follow the procedures for DF Election. In the proposed method which we will describe shortly this "jittered" behavior is retained.

5. HRW and Consistent Hashing

HRW is not the only algorithm that addresses the object to server mapping problem with goals of fair load distribution, redundancy and fast access. There is another family of algorithms that also addresses this problem; these fall under the umbrella of the Consistent Hashing Algorithms [CHASH]. These will not be considered here.

6. HRW Algorithm for EVPN DF Election

The applicability of HRW to DF Election can be described here. Let DF(v) denote the Designated Forwarder and BDF(v) the Backup Designated forwarder for the ethernet tag V, where v is the vlan, Si is the IP address of server i and weight is a pseudorandom function of v and Si

1. DF(v) = Si: Weight(v, Si) >= Weight(V, Sj) , for all j. In case of a tie, choose the PE whose IP address is numerically the least.

2. BDF(v) = Sk: Weight(v, Si) >= Weight(V, Sk) and Weight(v, Sk) >= Weight(v, Sj). in case of tie choose the PE whose IP address is numerically the least.

Since the Weight is a Pseudorandom function with domain as a concatenation of (v, S), it is an efficient deterministic algorithm which is independent of the Ethernet Tag V sample space distribution. Choosing a good hash function for the pseudorandom function is an important consideration for this algorithm to perform provably better than the existing algorithm. As mentioned previously, such functions are described in the HRW paper. We take as candidate hash functions two of the ones that are preferred in [HRW1999].

1. Wrand(v, Si) = (1103515245((1103515245.Si+12345)XOR D(v))+12345)(mod 2^31) and

2. Wrand2(v, Si) = (1103515245((1103515245.D(v)+12345)XOR Si)+12345)(mod 2^31)

Here D(v) is the 31-bit digest of the ethernet-tag v and Si is address of the ith server. The server's IP address length does not matter as only the low-order 31 bits are modulo significant.

A point to note is that the the domain of the Weight function is a concatenation of the ethernet-tag and the PE IP-address, and the actual length of the server IP address (whether V4 or V6) is not really relevant, so long as the actual hash algorithm takes into consideration the concatenated string. The existing algorithm in [RFC7432] as is cannot employ both V4 and V6 neighbor peering address.

HRW solves the disadvantage pointed out in Section 3 and ensures (i) with very high probability that the task of DF election for respective vlans is more or less equally distributed among the PEs even for the 2 PE case (ii)If a PE, hosting some vlans on given ES, but is neither the DF nor the BDF for that vlan, goes down or its connection to the ES goes down, it does not result in a DF and BDF reassignment the other PEs. This saves computation, especially in the case when the connection flaps. (iii)More importantly it avoids the needless disruption case (c) that are inherent in the existing modulus based algorithm (iv)In addition to the DF, the algorithm also furnishes the BDF, which would be the DF if the current DF fails.

7. Protocol Considerations

Note that 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 not possible that some PEs continue to use the existing modulus based DF election and some newer PEs use the HRW. For brownfield deployments and for interoperability with legacy boxes, its is important that all PEs need to have the capability to fall back on the modulus algorithm. A PE (one with a newer version of the software) can indicate its willingness to support HRW by signaling a new extended community along with the Ethernet-Segment Route (Type-4). This extended community is explained in the next paragraph. When a PE receives the Ethernet-Segment Routes from all the other PEs for the ethernet segment in question, it checks to see if all the advertisements have the extended community attached; in the case that they do, this particular PE, and by induction all the other PEs proceed to do DF Election as per the HRW Algorithm. Otherwise if even a single advertisement for the type-4 route is not received with the extended community, the default modulus algorithm is used as before. Also, the HRW algorithm needs to be executed after the "jittered" time.

A new BGP extended community attribute [RFC4360] needs to be defined to identify the DF election procedure to be used for the Ethernet Segment. We propose to name this extended community as the DF Election Extended Community. It is a new transitive extended community where the Type field is 0x06, and the Sub-Type is to be defined. It may be advertised along with Ethernet Segment routes.

Each DF Election Extended Community is encoded as a 8-octet value as follows:

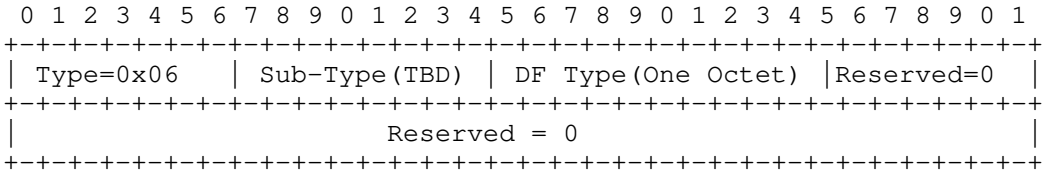


Figure 2

The DF Type state is encoded as one octet. A value of 0 means that the default (the mod based) DF election procedures are used and a value of 1 means that the HRW algorithm will be employed. A request needs to registered with the IETF authority for the subtype [I-D.ietf-idr-extcomm-iana]

8. Operational Considerations

TBD.

9. Security Considerations

This document raises no new security issues for EVPN.

10. Acknowledgements

The authors would like to thank Tamas Mondal and Sami Boutros for their feedback and useful discussions

11. References

11.1. Normative References

[HRW1999] Thaler, D. and C. Ravishankar, "Using Name-Based Mappings to Increase Hit Rates", IEEE/ACM Transactions in networking Volume 6 Issue 1, February 1998.

[I-D.ietf-idr-extcomm-iana] Rosen, E. and Y. Rekhter, "IANA Registries for BGP Extended Communities", draft-ietf-idr-extcomm-iana-02 (work in progress), December 2013.

[I-D.ietf-l2vpn-evpn]
Sajassi, A., Aggarwal, R., Bitar, N., Isaac, A., and J. Uttaro, "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-07 (work in progress), May 2014.

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

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, February 2006.

[RFC4761] Kompella, K. and Y. Rekhter, "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, January 2007.

[RFC7432] Sajassi, A., Aggarwal, R., Bitar, N., Isaac, A., Uttaro, J., Drake, J., and W. Henderickx, "BGP MPLS-Based Ethernet VPN", RFC 7432, February 2015.

11.2. Informative References

[CHASH] Karger, D., Lehman, E., Leighton, T., Panigrahy, R., Levine, M., and D. Lewin, "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web", ACM Symposium on Theory of Computing ACM Press New York, May 1997.

[CLRS2009]
Cormen, T., Leiserson, C., Rivest, R., and C. Stein, "Introduction to Algorithms (3rd ed.)", MIT Press and McGraw-Hill ISBN 0-262-03384-4., February 2009.

[RFC2991] Thaler, D. and C. Hopps, "Multipath Issues in Unicast and Multicast Next-Hop Selection", RFC 2991, November 2000.

[RFC2992] Hopps, C., "Analysis of an Equal-Cost Multi-Path Algorithm", RFC 2992, November 2000.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[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, November 2006.

[RFC6624] Kompella, K., Kothari, B., and R. Cherukuri, "Layer 2 Virtual Private Networks Using BGP for Auto-Discovery and Signaling", RFC 6624, May 2012.

Authors' Addresses

Satya Ranjan Mohanty
Cisco Systems, Inc.
225 West Tasman Drive
San Jose, CA 95134
USA

Email: satyamoh@cisco.com

Keyur Patel
Cisco Systems, Inc.
225 West Tasman Drive
San Jose, CA 95134
USA

Email: keyupate@cisco.com

Ali Sajassi
Cisco Systems, Inc.
225 West Tasman Drive
San Jose, CA 95134
USA

Email: sajassi@cisco.com

John Drake
Juniper Networks, Inc.
1194 N. Mathilda Drive
Sunnyvale, CA 95134
USA

Email: jdrake@juniper.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 26, 2015

T. Morin, Ed.
S. Litkowski
Orange
K. Patel
Cisco Systems
J. Zhang
R. Kebler
J. Haas
Juniper Networks

October 23, 2014

Multicast VPN state damping
draft-morin-bess-multicast-damping-01

Abstract

This document describes procedures to damp multicast VPN routing state changes and control the effect of the churn due to the multicast dynamicity in customer site. The procedures described in this document are applicable to BGP-based multicast VPN and help avoid uncontrolled control plane load increase in the core routing infrastructure. New procedures are proposed inspired from BGP unicast route damping principles, but adapted to multicast.

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].

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 April 26, 2015.

Table of Contents

1. Introduction

In a multicast VPN [RFC6513] deployed with BGP-based procedures [RFC6514], when receivers in VPN sites join and leave a said multicast group or channel through multicast membership control protocols (IGMP, MLD), multicast routing protocols accordingly adjust

multicast routing states and P-multicast tree states, to forward or prune multicast traffic to these receivers.

In VPN contexts, providing isolation between customers of a shared infrastructure is a core requirement resulting in stringent expectations with regards to risks of denial of service attacks. Hence, mechanisms need to be put in place to ensure that the load put on the BGP control plane, and on the P-tunnel setup control plane, remains under control regardless of the frequency at which multicast memberships changes are made by end hosts. By nature multicast memberships change based on the behavior of multicast applications running on end hosts, hence the frequency of membership changes can legitimately be much higher than the typical churn of unicast routing states. Section 16 of [RFC6514] specifically spells out the need for damping the activity of C-multicast and Leaf Auto-discovery routes.

This document describes procedures, remotely inspired from existing BGP route damping, aimed at protecting these control planes while at the same time avoiding negative effects on the service provided, although at the expense of a minimal increase in average of bandwidth use in the network.

The base principle is described in Section 3. Existing mechanisms that could be relied upon are discussed in Section 4. Section 5 details the procedures introduced by these specifications.

Section 6 provide specific details related to the damping of multicast VPNs P-tunnel state.

Finally, Section 7 discusses operational considerations related to the proposed mechanism.

2. Terminology

TBC

3. Overview

The procedures described in this document allows the network operator to configure multicast VPN PEs so that they can delay the propagation of multicast state prune messages, when faced with a rate of multicast state dynamicity exceeding a certain configurable threshold. Assuming that the number of multicast states that can be created by a receiver is bounded, delaying the propagation of multicast state pruning results in setting up an upper bound to the average frequency at which the router will send state updates to an upstream router.

From the point of view of a downstream router, such as a CE, this approach has no impact: the multicast routing states changes that it solicits to its PE will be honored without any additional delay. Indeed the propagation of joins is not impacted by the proposed defined procedures, and having the upstream router delay state prune propagation to its own upstream does not affect what traffic is sent to the downstream router. In particular, the amount of bandwidth used on the PE-CE link downstream to a PE applying this damping technique is not increased.

This approach increases the average bandwidth utilization on a link upstream to a PE applying this technique, such as a PE-PE link: indeed, a said multicast flow will be forwarded for a longer time than if no damping was applied. That said, it is expected that this technique will allow to meet the goals of protecting the multicast routing infrastructure control plane without a significant average increase of bandwidth; for instance, damping events happening at a frequency higher than one event per X second, can be done without increasing by more than X second the time during which a multicast flow is present on a link.

To be practical, such a mechanism requires configurability, in particular, needs to offer means to control when damping is triggered, and to allow delaying a multicast state Prune for a time increasing with the churn of this multicast state.

4. Existing mechanisms

This section describes mechanisms that could be considered to address the issue, but that end up appearing as not suitable or not efficient enough.

4.1. Rate-limiting of multicast control traffic

[RFC4609] examines multicast security threats and among other things the risk described in Section 1. A mechanism relying on rate-limiting PIM messages is proposed in section 5.3.3 [RFC4609], but has the identified drawbacks of impacting the service delivered and having side-effects on legitimate users.

4.2. Existing PIM, IGMP and MLD timers

In the context of PIM multicast routing protocols [RFC4601], a mechanism exists that in some context may offer a form of de facto damping mechanism for multicast states. Indeed, when active, the prune override mechanism consists in having a PIM upstream router introduce a delay ("prune override interval") before taking into account a PIM Prune message sent by a downstream neighbor.

This mechanism has not been designed specifically for the purpose of damping multicast state, but as a means to allow PIM to operate on multi-access networks. See [RFC4601] section 4.3.3. However, when active, this mechanism will prevent a downstream router to produce multicast routing protocol messages that would cause, for a said multicast state, the upstream router to send to its own upstreamrouter, multicast routing protocol messages at a rate higher than 1/[prune override interval], thus providing de-facto a form of damping.

Similarly, the IGMP and MLD multicast membership control protocols can provide a similar behavior, under the right conditions.

These mechanisms are not considered suitable to meet the goals spelled out in Section 1, the main reasons being that:

- when enabled these mechanisms require additional bandwidth on the local link on which the effect of a Prune is delayed (in our case the PE-CE link)
- when enabled these mechanisms require disabling explicit tracking, even though enabling this feature may otherwise be desired
- on certain implementations, these mechanisms are incompatible with behavior that cannot be turned off
- they do not provide a suitable level of configurability
- they do not provide a way to discriminate between multicast flows based on estimation of their dynamicity

4.3. BGP Route Damping

The procedures defined in [RFC2439] and [RFC7196] for BGP route flap damping are useful for operators who want to control the impact of unicast route churn on the routing infrastructure, and offer a standardized set of parameters to control damping.

These procedures are not directly relevant in a multicast context, for the following reasons:

- they are not specified for multicast routing protocol in general
- even in contexts where BGP routes are used to carry multicast routing states (e.g. [RFC6514]), these procedures do not allow to implement the principle described in this document, the main reason being that a damped route becomes suppressed, while the

target behavior would be to keep advertising when damping is triggered on a multicast route

However, the set of parameters standardized to control the thresholds of the exponential decay mechanism can be relevantly reused. This is the approach proposed for the procedures described in this document (Section 5). Motivations for doing so is to help the network operator deploy this feature based on consistent configuration parameter, and obtain predictable results, without the drawbacks of exposed in Section 4.1 and Section 4.2.

5. Procedures for multicast state damping

5.1. PIM procedures

This section describes procedures for multicast state damping satisfying the goals spelled out in Section 1. This section spells out procedures for (S,G) states in the PIM-SM protocol ([RFC4601] ; they apply unchanged for such states created based on multicast group management protocols (IGMP [RFC3376], MLD [RFC3810]) on downstream interfaces. The same procedures are applied to (*,G) states in the context of PIM-SM ASM groups (damping is not applied to (S,G,Rpt) Prune state).

The following notions introduced in [RFC2439] are reused in these procedures:

figure-of-merit: a number reflecting the current estimation of past recent activity of an (S,G) multicast routing state, which evolves based on routing events related to this state and based an exponential decay algorithm ; the activation or inactivation of damping on the state is based on this number ; this number is associated to the upstream state machine for (S,G)

cutoff-threshold: value of the *figure-of-merit* over which damping is applied (configurable parameter)

reuse-threshold: value of the *figure-of-merit* under which damping stops being applied (configurable parameter)

decay-half-life: period of time used to control how fast is the exponential decay of the *figure-of-merit* (configurable parameter)

Additionally to these values, a configurable "*increment-factor*" parameter is introduced, that controls by how much the *figure-of-merit* is incremented on multicast state update events.

Section Section 7.3 proposes default and maximum values for the configurable parameters.

On reception of updated multicast membership or routing information on a downstream interface I for a said (S,G) state, that results in a change of the state of the PIM downstream state machine (see section 4.5.3 of [RFC4601]), a router implementing these procedures MUST:

- apply unchanged procedures for everything relating to what multicast traffic ends up being sent on downstream interfaces, including interface I
- increasing the *figure-of-merit* for the (S,G) by the *increment-factor* (updating the *figure-of-merit* based on the decay algorithm must be done prior to this increment)
- update the damping state for the (S,G) state: damping becomes active on the state if the recomputed *figure-of-merit* is above the configured *cutoff-threshold*
- if damping is inactive on (S,G) state, update the upstream state machine as usual (as per section 4.5.7 of [RFC4601])
- if damping becomes active for the (S,G) state:
  - if the received message has caused the upstream state machine to transition to Joined state, update the upstream state machine for (S,G) (applying usual PIM procedures in section 4.5.7 of [RFC4601], including sending a PIM Join to the upstream neighbor)
  - if the received message has caused the upstream state machine to transition to NotJoined state, do not update the upstream state machine for (S,G)
  - then freeze the upstream state machine in Joined state, and and setup a trigger to update it once damping later becomes inactive again. The effect is that in the meantime, PIM Join messages will be sent as refreshes to the upstream neighbor, but no PIM Prune message will be sent.
- if damping was already active: do not update the upstream state machine for (S,G) (the upstream state machine was frozen after processing the previous message)

Once the *figure-of-merit* for (S,G) damping state decays to a value below the configured *reuse-threshold*, the upstream state machine for (S,G) is recomputed based on states of downstream state machines,

eventually leading to a PIM Join or Prune message to be sent to the upstream neighbor.

Same techniques as the ones described in [RFC2439] can be applied to determine when the figure-of-merit value is recomputed based on the exponential decay algorithm and the configured *decay-half-life*.

Given the specificity of multicast applications, it is REQUIRED for the implementation to let the operator configure the *decay-half-life* in seconds, rather than in minutes. When the recomputation is done periodically, the period should be low enough to not significantly delay the inactivation of damping on a multicast state beyond what the operator wanted to configure (i.e. for a half-life of 10s, recomputing the *figure-of-merit* each minute would result in a multicast state to remained damped for a much longer time than what the parameters are supposed to command).

PIM implementations typically follow [RFC4601] suggestion that "implementations will only maintain state when it is relevant to forwarding operations - for example, the 'NoInfo' state might be assumed from the lack of other state information, rather than being held explicitly" (Section 4.1 of [RFC4601]). To properly implement implement damping procedures, an implementation MUST keep an explicit (S,G) state as long as damping is active on an (S,G). Once an (S,G) state expires, and damping becomes inactive on this state, its associated *figure-of-merit* and damping state are removed as well.

Note that these procedures:

- do not impact PIM procedures related to refreshes or expiration of multicast routing states: PIM Prune messages triggered by the expiration of the (S,G) keep-alive timer, are not suppressed or delayed, and the reception of Join messages not causing transition of state on the downstream interface does not lead to incrementing the *figure-of-merit*;

- do not impact the PIM assert mechanism, in particular PIM Prune messages triggered by a change of the PIM assert winner on the upstream interface, are not suppressed or delayed;

- do not impact PIM Prune messages that are sent when the RPF neighbor is updated for a said multicast flow;

- do not impact PIM Prune messages that are sent in the context of switching between a Rendez-vous Point Tree and a Shortest Path Tree.

Note also that no action is triggered based on the reception of PIM Prune messages (or corresponding IGMP/MLD messages) that relate to non-existing (S,G) state, in particular, no *figure-of-merit* or damping state is created in this case.

5.2. Procedures for multicast VPN state dampening

The procedures described in Section 5.1 can be applied in the VRF PIM-SM implementation (in the "C-PIM instance"), with the corresponding action to suppressing the emission of a Prune(S,G) message being to not withdraw the C-multicast Source Tree Join (C-S,C-G) BGP route. Implementation of [RFC6513] relying on the use of PIM to carry C-multicast routing information MUST support this technique.

In the context of [RFC6514] where BGP is used to distribute C-multicast routing information, the following procedure is proposed as an alternative and consists in applying damping in the BGP implementation, based on existing BGP damping mechanism, applied to C-multicast Source Tree Join routes and Shared Tree Join routes (and as well to Leaf A-D routes - see Section 6), and modified to implement the behavior described in Section 3 along the following guidelines:

- not withdrawing (instead of not advertising) damped routes
- providing means to configure the half-life in seconds if that option is not already available
- using parameters for the exponential decay that are specific to multicast, based on default values and multicast specific configuration

While these procedures would typically be implemented on PE routers, in a context where BGP Route Reflectors are used it can be considered useful to also be able to apply damping on RRs as well. Additionally, for mVPN Inter-AS deployments, it can be needed to protect one AS from the dynamicity of multicast VPN routing events from other ASes. In that perspective, it is RECOMMENDED for implementations to support damping mVPN C-multicast routes directly into BGP, without relying on the PIM-SM state machine.

When not all routers in a deployment have the capability to drop traffic coming from the wrong PE (as spelled out in section 9.1.1 of [RFC6513]), then the withdrawal of a C-multicast route resulting from a change in the UMH SHOULD NOT be damped. An implementation of these specs MUST whether, not damp these withdrawals by default, or alternatively provide a tuning knob to disable then damping of these

withdrawals. Additionally, in such a context, it is RECOMMENDED to *not* enable any multicast VPN route damping on RRs and ASBRs, since these equipments cannot distinguish these events.

The choice to implement damping based on BGP routes or the procedures described in Section 5, is up to the implementor, but at least one of the two MUST be implemented; keeping in mind that in contexts where damping on RRs and ASBRs the BGP approach is RECOMMENDED.

Note well that damping SHOULD NOT be applied to BGP routes of the following sub-types: "Intra-AS I-PMSI A-D Route", "Inter-AS I-PMSI A-D Route", "S-PMSI A-D Route", and "Source Active A-D Route".

6. Procedures for P-tunnel state damping

6.1. Damping mVPN P-tunnel change events

When selective P-tunnels are used (see section 7 of [RFC6513]), the effect of updating the upstream state machine for a said (C-S,C-G) state on a PE connected to multicast receivers, is not only to generate activity to propagate C-multicast routing information to the source connected PE, but also to possibly trigger changes related to the P-tunnels carrying (C-S,C-G) traffic. Protecting the provider network from an excessive amount of change in the state of P-tunnels is required, and this section details how this can be done.

A PE implementing these procedures for mVPN MUST damp Leaf A-D routes, in the same manner as it would for C-multicast routes (see Section 5.2).

A PE implementing these procedures for mVPN MUST damp the activity related to removing itself from a P-tunnel. Possible ways to do so depend on the type of P-tunnel, and local implementation details are left up to the implementor.

The following is proposed as example of how the above can be achieved.

- For P-tunnels implemented with the PIM protocol, this consists in applying multicast state damping techniques described in Section 5.1 to the P-PIM instance, at least for (S,G) states corresponding to P-tunnels.

- For P-tunnels implemented with the mLDP protocol, this consists in applying damping techniques completely similar as the one described in Section 5, but generalized to apply to mLDP states

- For root-initiated P-tunnels (P-tunnels implemented with the P2MP RSVP-TE, or relying on ingress replication), no particular action needs to be implemented to damp P-tunnels membership, if the activity of Leaf A-D route themselves is damped

- Another possibility is to base the decision to join or not join the P-tunnel to which a said (C-S,C-G) is bound, and to advertise or not advertise a Leaf A-D route related to (C-S,C-G), based on whether or not a C-multicast Source Tree Join route is being advertised for (C-S,C-G), rather than by relying on the state of the C-PIM Upstream state machine for (C-S,C-G)

6.2. Procedures for Ethernet VPNs

Specifications exists to support or optimize multicast and broadcast in the context of Ethernet VPNs [RFC7117], relying on the use of S-PMSI and P-tunnels. For the same reasons as for IP multicast VPNs, an implementation of these procedures MUST follow the procedures described in this section.Section 6.1.

7. Operational considerations

7.1. Enabling and configuring multicast damping

In the context of multicast VPNs, these procedures would be enabled on PE routers. Additionally in the case of C-multicast routing based on BGP extensions ([RFC6514]) these procedures can be enabled on ASBRs, and possibly Route Reflectors as well.

7.2. Troubleshooting and monitoring

Implementing the damping mechanisms described in this document should be complemented by appropriate tools to observe and troubleshoot damping activity.

More specifically it is RECOMMENDED to complement the existing interface providing information on multicast states with information on eventual damping of corresponding states (e.g. MRIB states): C-multicast routing states and P-tunnel states.

7.3. Default and maximum values

The following values are RECOMMENDED to adopt as default conservative values:

- increment-factor: 1000

- cutoff-threshold: 3000

- decay-half-life: 10s
- reuse-threshold: 1500

For unicast damping, it is common to set an upper bound to the time during which a route is suppressed. In the case of multicast state damping, which relies on not withdrawing a damped route, it may be desirable to avoid a situation were a multicast flow would keep flowing in a portion of the network for a very large time in the absence of receivers.

The proposed default maximum value for the figure-of-merit is 20x<increment-factor>, i.e. 20000 with the proposed default increment-factor of 1000.

The following values are proposed as maximums:

- decay half-life: 60s
- cutoff-threshold: 50000

8. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

9. Security Considerations

The procedures defined in this document do not introduce additional security issues not already present in the contexts addressed, and actually aim at addressing some of the identified risks without introducing as much denial of service risk as some of the mechanisms already defined.

The protection provided relates to the control plane of the multicast routing protocols, including the components implementing the routing protocols and the components responsible for updating the multicast forwarding plane.

The procedures describe are meant to provide some level of protection for the router on which they are enabled by reducing the amount of routing state updates that it needs to send to its upstream neighbor or peers, but do not provide any reduction of the control plane load related to processing routing information from downstream neighbors. Protecting routers from an increase in control plane load due to activity on downstream interfaces toward core routers (or in the

context of BGP-based mVPN C-multicast routing, BGP peers) shall rely upon the activation of damping on corresponding downstream neighbors (or BGP peers) and/or at the edge of the network. Protecting routers from an increase in control plane load due to activity on customer-facing downstream interfaces or downstream interfaces to routers in another administrative domain, is out of the scope of this document and should rely upon already defined mechanisms (see [RFC4609]).

To be effective the procedures described here must be complemented by configuration limiting the number of multicast states that can be created on a multicast router through protocol interactions with multicast receivers, neighbor routers in adjacent ASes, or in multicast VPN contexts with multicast CEs. Note well that the two mechanism may interact: state for which Prune has been requested may still remain taken into account for some time if damping has been triggered and hence result in otherwise acceptable new state from being successfully created.

Additionally, it is worth noting that these procedures are not meant to protect against peaks of control plane load, but only address averaged load. For instance, assuming a set of multicast states submitted to the same Join/Prune events, damping can prevent more than a certain number of Join/Prune messages to be sent upstream in the period of time that elapses between the reception of Join/Prune messages triggering the activation of damping on these states and when damping becomes inactive after decay.

10. Acknowledgements

We would like to thank Bruno Decraene and Lenny Giuliano for discussions that helped shape this proposal. We would also like to thank Yakov Rekhter and Eric Rosen for their reviews and helpful comments. Thanks to Wim Henderickx for his comments and support of this proposal.

11. References

11.1. Normative References

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

[RFC2439] Villamizar, C., Chandra, R., and R. Govindan, "BGP Route Flap Damping", RFC 2439, November 1998.

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

[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.

[RFC7117] Aggarwal, R., Kamite, Y., Fang, L., Rekhter, Y., and C. Kodeboniya, "Multicast in Virtual Private LAN Service (VPLS)", RFC 7117, February 2014.

[RFC7196] Pelsser, C., Bush, R., Patel, K., Mohapatra, P., and O. Maennel, "Making Route Flap Damping Usable", RFC 7196, May 2014.

11.2. Informative References

[RFC4609] Savola, P., Lehtonen, R., and D. Meyer, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements", RFC 4609, October 2006.

Authors' Addresses

Thomas Morin (editor)
Orange
2, avenue Pierre Marzin
Lannion 22307
France

Email: thomas.morin@orange.com

Stephane Litkowski
Orange
France

Email: stephane.litkowski@orange.com

Keyur Patel
Cisco Systems
170 W. Tasman Drive
San Jose, CA 95134
USA

Email: keyupate@cisco.com

Jeffrey (Zhaohui) Zhang
Juniper Networks Inc.
10 Technology Park Drive
Westford, MA 01886
USA

Email: zzhang@juniper.net

Robert Kebler
Juniper Networks Inc.
10 Technology Park Drive
Westford, MA 01886
USA

Email: rkebler@juniper.net

Jeff Haas
Juniper Networks

Email: jhaas@juniper.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2016

T. Morin, Ed.
Orange
R. Kebler, Ed.
Juniper Networks

July 6, 2015

Multicast VPN fast upstream failover
draft-morin-bess-mvpn-fast-failover-02

Abstract

This document defines multicast VPN extensions and procedures that allow fast failover for upstream failures, by allowing downstream PEs to take into account the status of Provider-Tunnels (P-tunnels) when selecting the upstream PE for a VPN multicast flow, and extending BGP MVPN routing so that a C-multicast route can be advertized toward a standby upstream PE.

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].

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 3, 2016.

Copyright Notice

Table of Contents

1. Introduction

In the context of multicast in BGP/MPLS VPNs, it is desirable to provide mechanisms allowing fast recovery of connectivity on different types of failures. This document addresses failures of

elements in the provider network that are upstream of PEs connected to VPN sites with receivers.

The sections 3 and 4 describe two independent mechanisms, allowing different levels of resiliency, and providing different failure coverage:

- Section 3 describes local procedures allowing an egress PE (a PE connected to a receiver site) to take into account the status of P-Tunnels to determine the Upstream Multicast Hop (UMH) for a given (C-S, C-G). This method does not provide a "fast failover" solution when used alone, but can be used with the following sections for a "fast failover" solution.

- Section 4 describes protocol extensions that can speed up failover by not requiring any multicast VPN routing message exchange at recovery time.

Moreover, section 5 describes a "hot leaf standby" mechanism, that uses a combination of these two mechanisms. This approach has similarities with the solution described in [I-D.mofrr] to improve failover times when PIM routing is used in a network given some topology and metric constraints.

2. Terminology

The terminology used in this document is the terminology defined in [RFC6513] and [RFC6514].

3. UMH Selection based on tunnel status

Current multicast VPN specifications [RFC6513], section 5.1, describe the procedures used by a multicast VPN downstream PE to determine what the upstream multicast hop (UMH) is for a said (C-S,C-G).

The procedure described here is an OPTIONAL procedure that consists of having a downstream PE take into account the status of P-tunnels rooted at each possible upstream PEs, for including or not including each said PE in the list of candidate UMHs for a said (C-S,C-G) state. The result is that, if a P-tunnel is "down" (see Section 3.1), the PE that is the root of the P-Tunnel will not be considered for UMH selection, which will result in the downstream PE to failover to the upstream PE which is next in the list of candidates.

A downstream PE monitors the status of the tunnels of UMHs that are ahead of the current one. Whenever the downstream PE determines that

one of these tunnels is no longer "known to down", the PE selects the UMH corresponding to that as the new UMH.

More precisely, UMH determination for a said (C-S,C-G) will consider the UMH candidates in the following order:

- first, the UMH candidates that either (a) advertise a PMSI bound to a tunnel, where the specified tunnel is not known to be down or (b) do not advertise any I- or S- PMSI applicable to the said (C-S,C-G) but have associated a VRF Route Import BGP attribute to the unicast VPN route for S (this is necessary to avoid considering some invalid UMH PEs that use a policy where no I-PMSI is advertized for a said VRF and where only S-PMSI are used, the S-PMSI advertisement being possibly done only after the upstream PE receives a C-multicast route for (C-S, C-G)/(C-*, C-G) to be carried over the advertized S-PMSI)

- second, the UMH candidates that advertise a PMSI bound to a tunnel that is "down" -- these will thus be used as a last resort to ensure a graceful fallback to the basic MVPN UMH selection procedures in the hypothetical case where a false negative would occur when determining the status of all tunnels

For a said downstream PE and a said VRF, the P-tunnel corresponding to a said upstream PE for a said (C-S,C-G) state is the S-PMSI tunnel advertized by that upstream PE for this (C-S,C-G) and imported into that VRF, or if there isn't any such S-PMSI, the I-PMSI tunnel advertized by that PE and imported into that VRF.

Note that this documents assumes that if a site of a given MVPN that contains C-S is dual-homed to two PEs, then all the other sites of that MVPN would have two unicast VPN routes (VPN-IPv4 or VPN-IPv6) routes to C-S, each with its own RD.

3.1. Determining the status of a tunnel

Different factors can be considered to determine the "status" of a P-tunnel and are described in the following sub-sections. The procedure proposed here also allows that all downstream PEs don't apply the same rules to define what the status of a P-tunnel is (please see Section 6), and some of them will produce a result that may be different for different downstream PEs. Thus what is called the "status" of a P-tunnel in this section, is not a characteristic of the tunnel in itself, but is the status of the tunnel, *as seen from a particular downstream PE*. Additionally, some of the following methods determine the ability of downstream PE to receive traffic on the P-tunnel and not specifically on the status of the P-tunnel itself. This could be referred to as "P-tunnel reception

status", but for simplicity, we will use the terminology of P-tunnel "status" for all of these methods.

Depending on the criteria used to determine the status of a P-tunnel, there may be an interaction with another resiliency mechanism used for the P-tunnel itself, and the UMH update may happen immediately or may need to be delayed. Each particular case is covered in each separate sub-section below.

3.1.1. mVPN tunnel root tracking

A condition to consider that the status of a P-tunnel is up is that the root of the tunnel, as determined in the PMSI tunnel attribute, is reachable through unicast routing tables. In this case the downstream PE can immediately update its UMH when the reachability condition changes.

This is similar to BGP next-hop tracking for VPN routes, except that the address considered is not the BGP next-hop address, but the root address in the PMSI tunnel attribute.

If BGP next-hop tracking is done for VPN routes, and the root address of a said tunnel happens to be the same as the next-hop address in the BGP autodiscovery route advertising the tunnel, then this mechanisms may be omitted for this tunnel, as it will not bring any specific benefit.

3.1.2. PE-P Upstream link status

A condition to consider a tunnel status as up can be that the last-hop link of the P-tunnel is up.

This method should not be used when there is a fast restoration mechanism (such as MPLS FRR [RFC4090]) in place for the link.

3.1.3. P2MP RSVP-TE tunnels

For P-Tunnels of type P2MP MPLS-TE, the status of the P-Tunnel is considered up if one or more of the P2MP RSVP-TE LSPs, identified by the P-Tunnel Attribute, are in up state. The determination of whether a P2MP RSVP-TE LSP is in up state requires Path and Resv state for the LSP and is based on procedures in [RFC4875]. In this case the downstream PE can immediately update its UMH when the reachability condition changes.

When signaling state for a P2MP TE LSP is removed (e.g. if the ingress of the P2MP TE LSP sends a PathTear message) or the P2MP TE LSP changes state from up to down as determined by procedures in

[RFC4875], the status of the corresponding P-Tunnel SHOULD be re-evaluated. If the P-Tunnel transitions from up to down state, the upstream PE, that is the ingress of the P-Tunnel, SHOULD not be considered a valid UMH.

3.1.4. Leaf-initiated P-tunnels

A PE can be removed from the UMH candidate list for a said (S,G) if the P-tunnel for this S,G (I or S , depending) is leaf triggered (PIM, mLDP), but for some reason internal to the protocol the upstream one-hop branch of the tunnel from P to PE cannot be built. In this case the downstream PE can immediately update its UMH when the reachability condition changes.

3.1.5. (S,G) counter information

In cases, where the downstream node can be configured so that the maximum inter-packet time is known for all the multicast flows mapped on a P-tunnel, the local per-(C-S,C-G) traffic counter information for traffic received on this P-tunnel can be used to determine the status of the P-tunnel.

When such a procedure is used, in context where fast restoration mechanisms are used for the P-tunnels, downstream PEs should be configured to wait before updating the UMH, to let the P-tunnel restoration mechanism happen. A configurable timer MUST be provided for this purpose, and it is recommended to provide a reasonable default value for this timer.

This method can be applicable for instance when a (S,G) flow is mapped on an S-PMSI.

In cases where this mechanism is used in conjunction with Hot leaf standby, then no prior knowledge of the rate of the multicast streams is required ; downstream PEs can compare reception on the two P-tunnels to determine when one of them is down.

3.1.6. BFD Discriminator

P-tunnel status can be derived from the status of a BFD session whose discriminator is advertized along with an x-PMSI A-D route.

3.1.6.1. Root PE Procedures

When it is desired to track the P-Tunnel status using BFD, the Root PE MUST include the BGP-BFD Attribute in the x-PMSI A-D Route.

If a P-Tunnel is already signaled, and then it is desired to track the P-Tunnel status using BFD, x-PMSI A-D Route must be re-sent with the same attributes as before, but the BGP-BFD Attribute MUST be included.

If P-Tunnel is already signaled, and P-Tunnel status tracked using BFD and it is desired to stop tracking P-Tunnel status using BFD, then x-PMSI A-D Route MUST be re-sent with the same attributes as before, but the BGP-BFD Attribute MUST be excluded.

3.1.6.2. Leaf PE Procedures

On receiving the BFD attribute in the x-PMSI A-D Route, the Leaf PE MUST associate the received discriminator with the P-Tunnel originating from the Root PE. Once the Leaf PE start getting the BFD probes from the Root PE with the said discriminator, the BFD session will be declared up and will then be used to track the health of the P-Tunnel.

If the Leaf PE does not receive BFD probes for a P-Tunnel from the Root PE for Detection Time, the BFD session would be brought down. And, it would declare the P-tunnel associated with the discriminator as down.

Leaf PE then can then initiate a switchover of the traffic from the Primary Tunnel, to the Standby Tunnel.

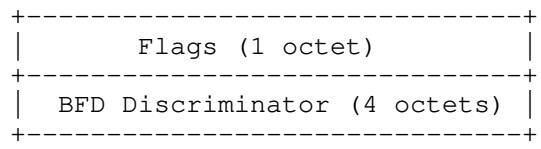
When Leaf PE's P-Tunnel is already up, it receives new x-PMSI A-D Route with BGP-BFD attribute, it must accept the x-PMSI A-D Route and associate the discriminator with the P-tunnel. When the BFD probes are received with the said discriminator, the BFD session is declared up.

When Leaf PE's P-Tunnel is already up, and is tracked with BFD, and it receives new x-PMSI A-D Route without BGP-BFD attribute, it must accept the x-PMSI A-D Route the BFD session should be declared admin down. Receiver node SHOULD not switch the traffic to the Standby P-tunnel.

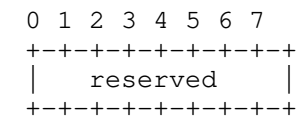
When such a procedure is used, in context where fast restoration mechanisms are used for the P-tunnels, leaf PEs should be configured to wait before updating the UMH, to let the P-tunnel restoration mechanism happen. A configurable timer MUST be provided for this purpose, and it is recommended to provide a reasonable default value for this timer.

3.1.6.3. BGP-BFD Attribute

This document defines and uses a new BGP attribute called the "BGP-BFD attribute". This is an optional transitive BGP attribute. The format of this attribute is defined as follows:



The Flags field has the following format:



3.1.7. Per PE-CE link BFD Discriminator

The following approach is proposed for fast failover on PE-CE link failures, in which UMH selection for a said (S,G) takes into account the state of a BFD session dedicated to the state of the upstream PE-CE link.

If this approach is enabled:

- each upstream PE: for each PE-CE link for which this protection is wanted, initiates a multipoint BFD session toward downstream PEs, with a trigger causing such a session to be torn down if the associated PE-CE link is detected as down.

- each upstream PE: for each prefix of a PE-CE link for which protection is wanted, advertizes a wildcard S-PMSI covering the sources inside this prefix, and signals along with this S-PMSI the multipoint BFD session discriminator associated with the PE-CE link. (note that all these S-PMSIs can perfectly use the same P-tunnel)

- each downstream PE: if an S-PMSI bound to a said (S,G) is signaled with a multipoint BFD session, then the upstream PE is considered during UMH selection for (S,G) if and only if the corresponding BFD session is up. Whenever the BFD session goes down the S-PMSI P-tunnel will be considered down and the downstream PE will switch to the backup P-tunnel. Note that the P-tunnel is considered down only for the (S,G) states that match to an S-PMSI signaling the BFD discriminator of a BFD session which is down

4. Standby C-multicast route

The procedures described below are limited to the case where the site that contains C-S is connected to exactly two PEs. The procedures require all the PEs of that MVPN to follow the single forwarder PE selection, as specified in [RFC6513]. The procedures assume that if a site of a given MVPN that contains C-S is dual-homed to two PEs, then all the other sites of that MVPN would have two unicast VPN routes (VPN-IPv4 or VPN-IPv6) routes to C-S, each with its own RD.

As long as C-S is reachable via both PEs, a said downstream PE will select one of the PEs connected to C-S as its Upstream PE with respect to C-S. We will refer to the other PE connected to C-S as the "Standby Upstream PE". Note that if the connectivity to C-S through the Primary Upstream PE becomes unavailable, then the PE will select the Standby Upstream PE as its Upstream PE with respect to C-S.

For readability, in the following sub-sections, the procedures are described for BGP C-multicast Source Tree Join routes, but they apply equally to BGP C-multicast Shared Tree Join routes failover for the case where the customer RP is dual-homed (substitute "C-RP" to "C-S").

4.1. Downstream PE behavior

When a (downstream) PE connected to some site of an MVPN needs to send a C-multicast route (C-S, C-G), then following the procedures specified in Section "Originating C-multicast routes by a PE" of [RFC6514] the PE sends the C-multicast route with RT that identifies the Upstream PE selected by the PE originating the route. As long as C-S is reachable via the Primary Upstream PE, the Upstream PE is the Primary Upstream PE. If C-S is reachable only via the Standby Upstream PE, then the Upstream PE is the Standby Upstream PE.

If C-S is reachable via both the Primary and the Standby Upstream PE, then in addition to sending the C-multicast route with an RT that identifies the Primary Upstream PE, the PE also originates and sends a C-multicast route with an RT that identifies the Standby Upstream

PE. This route, that has the semantic of being a 'standby' C-multicast route, is further called a "Standby BGP C-multicast route", and is constructed as follows:

- the NLRI is constructed as the original C-multicast route, except that the RD is the same as if the C-multicast route was built using the standby PE as the UMH (it will carry the RD associated to the unicast VPN route advertized by the standby PE for S)

- SHOULD carry the "Standby PE" BGP Community (this is a new BGP Community, see Section 7)

The normal and the standby C-multicast routes must have their Local Preference attribute adjusted so that, if two C-multicast routes with same NLRI are received by a BGP peer, one carrying the "Standby PE" attribute and the other one *not* carrying the "Standby PE" community, then preference is given to the one *not* carrying the "Standby PE" attribute. Such a situation can happen when, for instance due to transient unicast routing inconsistencies, two different downstream PEs consider different upstream PEs to be the primary one ; in that case, without any precaution taken, both upstream PEs would process a standby C-multicast route and possibly stop forwarding at the same time. For this purpose a Standby BGP C-multicast route MUST have the LOCAL_PREF attribute set to zero.

Note that, when a PE advertizes such a Standby C-multicast join for an (S,G) it must join the corresponding P-tunnel.

If at some later point the local PE determines that C-S is no longer reachable through the Primary Upstream PE, the Standby Upstream PE becomes the Upstream PE, and the local PE re-sends the C-multicast route with RT that identifies the Standby Upstream PE, except that now the route does not carry the Standby PE BGP Community (which results in replacing the old route with a new route, with the only difference between these routes being the presence/absence of the Standby PE BGP Community).

4.2. Upstream PE behavior

When a PE receives a C-multicast route for a particular (C-S, C-G), and the RT carried in the route results in importing the route into a particular VRF on the PE, if the route carries the Standby PE BGP Community, then the PE performs as follows:

when the PE determines that C-S is not reachable through some other PE, the PE SHOULD install VRF PIM state corresponding to this Standby BGP C-multicast route (the result will be that a PIM Join message will be sent to the CE towards C-S, and that the PE

will receive (C-S,C-G) traffic), and the PE SHOULD forward (C-S, C-G) traffic received by the PE to other PEs through a P-tunnel rooted at the PE.

Furthermore, irrespective of whether C-S carried in that route is reachable through some other PE:

a) based on local policy, as soon as the PE receives this Standby BGP C-multicast route, the PE MAY install VRF PIM state corresponding to this BGP Source Tree Join route (the result will be that Join messages will be sent to the CE toward C-S, and that the PE will receive (C-S,C-G) traffic)

b) based on local policy, as soon as the PE receives this Standby BGP C-multicast route, the PE MAY forward (C-S, C-G) traffic to other PEs through a P-tunnel independently of the reachability of C-S through some other PE. [note that this implies also doing (a)]

Doing neither (a), nor (b) for a said (C-S,C-G) is called "cold root standby".

Doing (a) but not (b) for a said (C-S,C-G) is called "warm root standby".

Doing (b) (which implies also doing (a)) for a said (C-S,C-G) is called "hot root standby".

Note that, if an upstream PE uses an S-PMSI only policy, it shall advertise an S-PMSI for an (S,G) as soon as it receives a C-multicast route for (S,G), normal or Standby ; i.e. it shall not wait for receiving a non-Standby C-multicast route before advertising the corresponding S-PMSI.

Section 9.3.2 of [RFC6514], describes the procedures of sending a Source-Active A-D result as a result of receiving the C-multicast route. These procedures should be followed for both the normal and Standby C-multicast routes.

4.3. Reachability determination

The standby PE can use the following information to determine that C-S can or cannot be reached through the primary PE:

- presence/absence of a unicast VPN route toward C-S
- supposing that the standby PE is an egress of the tunnel rooted at the Primary PE, the standby PE can determine the reachability of C-S through the Primary PE based on the status of this tunnel,

determined thanks to the same criteria as the ones described in Section 3.1 (without using the UMH selection procedures of Section 3)

- other mechanisms MAY be used

4.4. Inter-AS

If the non-segmented inter-AS approach is used, the procedures in section 4 can be applied.

When multicast VPNs are used in a inter-AS context with the segmented inter-AS approach described in section 8.2 of [RFC6514], the procedures in this section can be applied.

A pre-requisite for the procedures described below to be applied for a source of a said MVPN is:

- that any PE of this MVPN receives two Inter-AS I-PMSI auto-discovery routes advertized by the AS of the source (or more)

- that these Inter-AS I-PMSI autodiscovery routes have distinct Route Distinguishers (as described in item "(2)" of section 9.2 of [RFC6514]).

As an example, these conditions will be satisfied when the source is dual homed to an AS that connects to the receiver AS through two ASBR using auto-configured RDs.

4.4.1. Inter-AS procedures for downstream PEs, ASBR fast failover

The following procedure is applied by downstream PEs of an AS, for a source S in a remote AS.

Additionally to choosing an Inter-AS I-PMSI autodiscovery route advertized from the AS of the source to construct a C-multicast route, as described in section 11.1.3 [RFC6514] a downstream PE will choose a second Inter-AS I-PMSI autodiscovery route advertized from the AS of the source and use this route to construct and advertise a Standby C-multicast route (C-multicast route carrying the Standby extended community) as described in Section 4.1.

4.4.2. Inter-AS procedures for ASBRs

When an upstream ASBR receives a C-multicast route, and at least one of the RTs of the route matches one of the ASBR Import RT, the ASBR locates an Inter-AS I-PMSI A-D route whose RD and Source AS matches the RD and Source AS carried in the C-multicast route. If the match

is found, and C-multicast route carries the Standby PE BGP Community, then the ASBR performs as follows:

- o if the route was received over iBGP ; the route is expected to have a LOCAL_PREF attribute set to zero and it should be re-advertized in eBGP with a MED attribute (MULTI_EXIT_DISC) set to the highest possible value (0xffff)

- o if the route was received over eBGP ; the route is expected to have a MED attribute set of 0xffff and should be re-advertized in iBGP with a LOCAL_PREF attribute set to zero

Other ASBR procedures are applied without modification.

5. Hot leaf standby

The mechanisms defined in sections Section 4 and Section 3 can be used together as follows.

The principle is that, for a said VRF (or possibly only for a said C-S,C-G):

- o downstream PEs advertise a Standby BGP C-multicast route (based on Section 4)

- o upstream PEs use the "hot standby" optional behavior and thus will forward traffic for a said multicast state as soon as they have whether a (primary) BGP C-multicast route or a Standby BGP C-multicast route for that state (or both)

- o downstream PEs accept traffic from the primary or standby tunnel, based on the status of the tunnel (based on Section 3)

Other combinations of the mechanisms proposed in Section 4) and Section 3 are for further study.

Note that the same level of protection would be achievable with a simple C-multicast Source Tree Join route advertized to both the primary and secondary upstream PEs (carrying as Route Target extended communities, the values of the VRF Route Import attribute of each VPN route from each upstream PEs). The advantage of using the Standby semantic for is that, supposing that downstream PEs always advertise a Standby C-multicast route to the secondary upstream PE, it allows to choose the protection level through a change of configuration on the secondary upstream PE, without requiring any reconfiguration of all the downstream PEs.

6. Duplicate packets

Multicast VPN specifications [RFC6513] impose that a PE only forwards to CEs the packets coming from the expected usptream PE (Section 9.1).

We highlight the reader's attention to the fact that the respect of this part of multicast VPN specifications is especially important when two distinct upstream PEs are susceptible to forward the same traffic on P-tunnels at the same time in steady state. This will be the case when "hot root standby" mode is used (Section 4), and which can also be the case if procedures of Section 3 are used and (a) the rules determining the status of a tree are not the same on two distinct downstream PEs or (b) the rule determining the status of a tree depend on conditions local to a PE (e.g. the PE-P upstream link being up).

7. IANA Considerations

Allocation is expected from IANA for the BGP "Standby PE" community. (TBC)

[Note to RFC Editor: this section may be removed on publication as an RFC.]

8. Security Considerations

9. Acknowledgements

The authors want to thank Greg Reaume and Eric Rosen for their review and useful feedback.

10. Contributor Addresses

Below is a list of other contributing authors in alphabetical order:

Rahul Aggarwal
Arktan

Email: raggarwa_1@yahoo.com

Nehal Bhau
Alcatel-Lucent, Inc.
701 E Middlefield Rd
Mountain View, CA 94043
USA

Email: Nehal.Bhau@alcatel-lucent.com

Clayton Hassen
Bell Canada
2955 Virtual Way
Vancouver
CANADA

Email: Clayton.Hassen@bell.ca

Wim Henderickx
Alcatel-Lucent
Copernicuslaan 50
Antwerp 2018
Belgium

Email: wim.henderickx@alcatel-lucent.com

Pradeep Jain
Alcatel-Lucent, Inc.
701 E Middlefield Rd
Mountain View, CA 94043
USA

Email: pradeep.jain@alcatel-lucent.com

Jayant Kotalwar
Alcatel-Lucent, Inc.
701 E Middlefield Rd
Mountain View, CA 94043
USA

Email: Jayant.Kotalwar@alcatel-lucent.com

Praveen Muley
Alcatel-Lucent
701 East Middlefield Rd
Mountain View, CA 94043
U.S.A.

Email: praveen.muley@alcatel-lucent.com

Ray (Lei) Qiu
Juniper Networks
1194 North Mathilda Ave.
Sunnyvale, CA 94089
U.S.A.

Email: rqiu@juniper.net

Yakov Rekhter
Juniper Networks
1194 North Mathilda Ave.
Sunnyvale, CA 94089
U.S.A.

Email: yakov@juniper.net

Kanwar Singh
Alcatel-Lucent, Inc.
701 E Middlefield Rd
Mountain View, CA 94043
USA

Email: kanwar.singh@alcatel-lucent.com

11. References

11.1. Normative References

[I-D.ietf-bfd-multipoint]
Katz, D., Ward, D., and S. Pallagatti, "BFD for Multipoint Networks", draft-ietf-bfd-multipoint-06 (work in progress), January 2015.

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

[RFC4875] Aggarwal, R., Papadimitriou, D., and S. Yasukawa, "Extensions to Resource Reservation Protocol - Traffic Engineering (RSVP-TE) for Point-to-Multipoint TE Label Switched Paths (LSPs)", RFC 4875, May 2007.

[RFC6513] Aggarwal, R., Bandi, S., Cai, Y., Morin, T., Rekhter, Y., Rosen, E., Wijnands, I., and S. Yasukawa, "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.

11.2. Informative References

[I-D.mofrr] Karan, A., Filsfils, C., Farinacci, D., Decraene, B., Leymann, N., and T. Telkamp, "Multicast only Fast Re-Route", draft-ietf-rtgwg-mofrr-08 (work in progress), February 2015.

[RFC4090] Pan, P., Swallow, G., and A. Atlas, "Fast Reroute Extensions to RSVP-TE for LSP Tunnels", RFC 4090, May 2005.

Authors' Addresses

Thomas Morin (editor)
Orange
2, avenue Pierre Marzin
Lannion 22307
France

Email: thomas.morin@orange-ftgroup.com

Robert Kebler (editor)
Juniper Networks
1194 North Mathilda Ave.
Sunnyvale, CA 94089
U.S.A.

Email: rkebler@juniper.net

BESS Workgroup
Internet Draft
Intended status: Standards Track

J. Rabadan
S. Sathappan
W. Henderickx
S. Palislamovic
Alcatel-Lucent

R. Shekhar
A. Lohiya
Juniper

F. Balus
Nuage Networks

A. Sajassi
D. Cai
Cisco

Expires: April 27, 2015 October 24, 2014

Interconnect Solution for EVPN Overlay networks
draft-rabadan-bess-dci-evpn-overlay-00

Abstract

This document describes how Network Virtualization Overlay networks (NVO) can be connected to a Wide Area Network (WAN) in order to extend the layer-2 connectivity required for some tenants. The solution analyzes the interaction between NVO networks running EVPN and other L2VPN technologies used in the WAN, such as VPLS/PBB-VPLS or EVPN/PBB-EVPN, and proposes a solution for the interworking between both.

Table of Contents

1. Introduction

[EVPN-Overlays] discusses the use of EVPN as the control plane for Network Virtualization Overlay (NVO) networks, where VXLAN, NVGRE or MPLS over GRE can be used as possible data plane encapsulation options.

While this model provides a scalable and efficient multi-tenant solution within the Data Center, it might not be easily extended to the WAN in some cases due to the requirements and existing deployed technologies. For instance, a Service Provider might have an already deployed (PBB-)VPLS or (PBB-)EVPN network that must be used to interconnect Data Centers and WAN VPN users. A Gateway (GW) function is required in these cases.

This document describes a Interconnect solution for EVPN overlay networks, assuming that the NVO Gateway (GW) and the WAN Edge functions can be decoupled in two separate systems or integrated into the same system. The former option will be referred as "Decoupled Interconnect solution" throughout the document whereas the latter one will be referred as "Integrated Interconnect solution".

2. Decoupled Interconnect solution for EVPN overlay networks

This section describes the interconnect solution when the GW and WAN Edge functions are implemented in different systems. Figure 1 depicts the reference model described in this section.

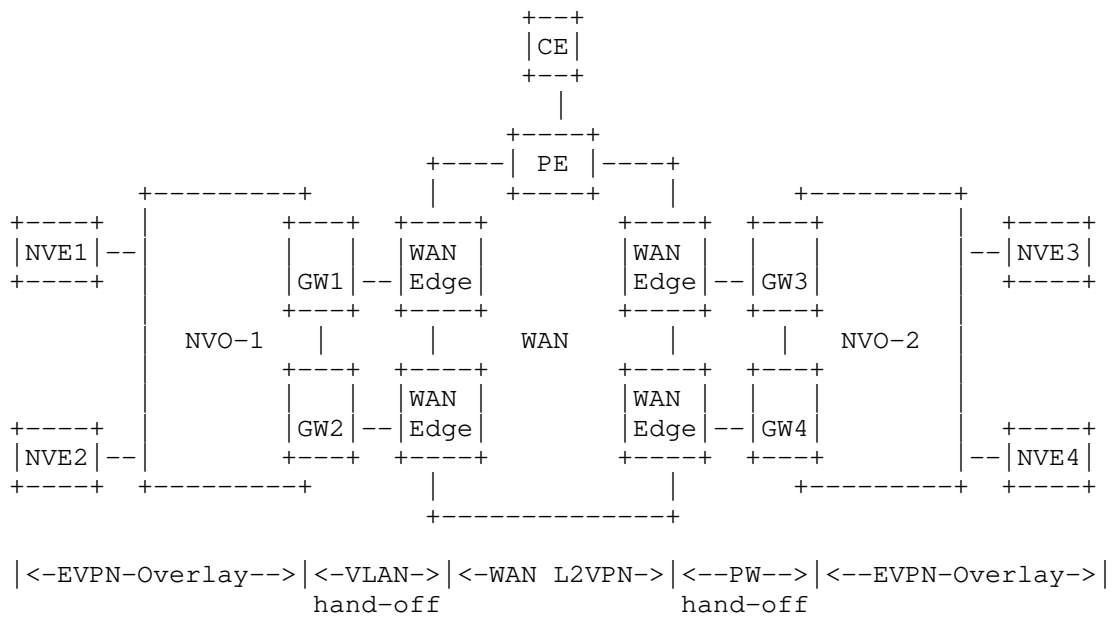


Figure 1 Decoupled Interconnect model

The following section describes the interconnect requirements for this model.

2.1. Interconnect requirements

This proposed Interconnect architecture will be normally deployed in networks where the EVPN-Overlay and WAN providers are different entities and a clear demarcation is needed. The solution must observe the following requirements:

- o A simple connectivity hand-off must be provided between the EVPN-Overlay network provider and the WAN provider so that QoS and security enforcement are easily accomplished.

- o The solution must be independent of the L2VPN technology deployed in the WAN.

- o Multi-homing between GW and WAN Edge routers is required. Per-service load balancing MUST be supported. Per-flow load balancing MAY be supported but it is not a strong requirement since a deterministic path per service is usually required for an easy QoS and security enforcement.

- o Ethernet OAM and Connectivity Fault Management (CFM) functions must

be supported between the EVPN-Overlay network and the WAN network.

- o The following optimizations MAY be supported at the GW:
  - + Flooding reduction of unknown unicast traffic sourced from the DC Network Virtualization Edge devices (NVEs).
  - + Control of the WAN MAC addresses advertised to the DC.
  - + ARP flooding control for the requests coming from the WAN.

2.2. VLAN-based hand-off

In this option, the hand-off between the GWs and the WAN Edge routers is based on 802.1Q VLANs. This is illustrated in Figure 1 (between the GWs in NVO-1 and the WAN Edge routers). Each MAC-VRF in the GW is connected to a different VSI/MAC-VRF instance in the WAN Edge router by using a different C-TAG VLAN ID or a different combination of S/C-TAG VLAN IDs that matches at both sides.

This option provides the best possible demarcation between the DC and WAN providers and it does not require control plane interaction between both providers. The disadvantage of this model is the provisioning overhead since the service must be mapped to a S/C-TAG VLAN ID combination at both, GW and WAN Edge routers.

In this model, the GW acts as a regular Network Virtualization Edge (NVE) towards the DC. Its control plane, data plane procedures and interactions are described in [EVPN-Overlays].

The WAN Edge router acts as a (PBB-)VPLS or (PBB-)EVPN PE with attachment circuits (ACs) to the GWs. Its functions are described in [RFC4761][RFC4762][RFC6074] or [EVPN][PBB-EVPN].

2.3. PW-based (Pseudowire-based) hand-off

If MPLS can be enabled between the GW and the WAN Edge router, a PW-based Interconnect solution can be deployed. In this option the hand-off between both routers is based on FEC128-based PWs or FEC129-based PWs (for a greater level of network automation). Note that this model still provides a clear demarcation boundary between DC and WAN, and security/QoS policies may be applied on a per PW basis. This model provides better scalability than a C-TAG based hand-off and less provisioning overhead than a combined C/S-TAG hand-off. The PW-based hand-off interconnect is illustrated in Figure 1 (between the NVO-2 GWs and the WAN Edge routers).

In this model, besides the usual MPLS procedures between GW and WAN Edge router, the GW MUST support an interworking function in each MAC-VRF that requires extension to the WAN:

- If a FEC128-based PW is used between the MAC-VRF (GW) and the VSI (WAN Edge), the provisioning of the VCID for such PW MUST be supported on the MAC-VRF and must match the VCID used in the peer VSI at the WAN Edge router.

- If BGP Auto-discovery [RFC6074] and FEC129-based PWs are used between the GW MAC-VRF and the WAN Edge VSI, the provisioning of the VPLS-ID MUST be supported on the MAC-VRF and must match the VPLS-ID used in the WAN Edge VSI.

2.4. Multi-homing solution on the GWs

As already discussed, single-active multi-homing, i.e. per-service load-balancing multi-homing MUST be supported in this type of interconnect. All-active multi-homing may be considered in future revisions of this document.

The GWs will be provisioned with a unique ESI per WAN interconnect and the hand-off attachment circuits or PWs between the GW and the WAN Edge router will be assigned to such ESI. The ESI will be administratively configured on the GWs according to the procedures in [EVPN]. This Interconnect ESI will be referred as "I-ESI" hereafter.

The solution (on the GWs) MUST follow the single-active multi-homing procedures as described in [EVPN-Overlays] for the provisioned I-ESI, i.e. Ethernet A-D routes per ESI and per EVI will be advertised to the DC NVEs. The MAC addresses learnt (in the data plane) on the hand-off links will be advertised with the I-ESI encoded in the ESI field.

2.5. Gateway Optimizations

The following features MAY be supported on the GW in order to optimize the control plane and data plane in the DC.

2.5.1 Use of the Unknown MAC route to reduce unknown flooding

The use of EVPN in the NVO networks brings a significant number of benefits as described in [EVPN-Overlays]. There are however some potential issues that SHOULD be addressed when the DC EVIs are connected to the WAN VPN instances.

The first issue is the additional unknown unicast flooding created in the DC due to the unknown MACs existing beyond the GW. In virtualized DCs where all the MAC addresses are learnt in the control/management plane, unknown unicast flooding is significantly reduced. This is no longer true if the GW is connected to a layer-2 domain with data plane learning.

The solution suggested in this document is based on the use of an "Unknown MAC route" that is advertised by the Designated Forwarder GW. The Unknown MAC route is a regular EVPN MAC/IP Advertisement route where the MAC Address Length is set to 48 and the MAC address to 00:00:00:00:00:00 (IP length is set to 0).

If this procedure is used, when an EVI is created in the GWs and the Designated Forwarder (DF) is elected, the DF will send the Unknown MAC route. The NVEs supporting this concept will prune their unknown unicast flooding list and will only send the unknown unicast packets to the owner of the Unknown MAC route. Note that the I-ESI will be encoded in the ESI field of the NLRI so that regular multi-homing procedures can be applied to this unknown MAC too (e.g. backup-path).

2.5.2. MAC address advertisement control

Another issue derived from the EVI interconnect to the WAN layer-2 domain is the potential massive MAC advertisement into the DC. All the MAC addresses learnt from the WAN on the hand-off attachment circuits or PWs must be advertised by BGP EVPN. Even if optimized BGP techniques like RT-constraint are used, the amount of MAC addresses to advertise or withdraw (in case of failure) from the GWs can be difficult to control and overwhelming for the DC network, especially when the NVEs reside in the hypervisors.

This document proposes the addition of administrative options so that the user can enable/disable the advertisement of MAC addresses learnt from the WAN as well as the advertisement of the Unknown MAC route from the DF GW. In cases where all the DC MAC addresses are learnt in the control/management plane, the GW may disable the advertisement of WAN MAC addresses. Any frame with unknown destination MAC will be exclusively sent to the Unknown MAC route owner(s).

2.5.3. ARP flooding control

Another optimization mechanism, naturally provided by EVPN in the GWs, is the Proxy ARP/ND function. The GWs SHOULD build a Proxy ARP/ND cache table as per [EVPN]. When the active GW receives an ARP/ND request/solicitation coming from the WAN, the GW does a Proxy ARP/ND table lookup and replies as long as the information is available in its table.

This mechanism is especially recommended on the GWs since it protects the DC network from external ARP/ND-flooding storms.

2.5.4. Handling failures between GW and WAN Edge routers

Link/PE failures MUST be handled on the GWs as specified in [EVPN].

The GW detecting the failure will withdraw the EVPN routes as per [EVPN].

Individual AC/PW failures should be detected by OAM mechanisms. For instance:

- If the Interconnect solution is based on a VLAN hand-off, 802.1ag/Y.1731 Ethernet-CFM MAY be used to detect individual AC failures on both, the GW and WAN Edge router. An individual AC failure will trigger the withdrawal of the corresponding A-D per EVI route as well as the MACs learnt on that AC.

- If the Interconnect solution is based on a PW hand-off, the LDP PW Status bits TLV MAY be used to detect individual PW failures on both, the GW and WAN Edge router.

3. Integrated Interconnect solution for EVPN overlay networks

When the DC and the WAN are operated by the same administrative entity, the Service Provider can decide to integrate the GW and WAN Edge PE functions in the same router for obvious CAPEX and OPEX saving reasons. This is illustrated in Figure 2. Note that this model does not provide an explicit demarcation link between DC and WAN anymore.

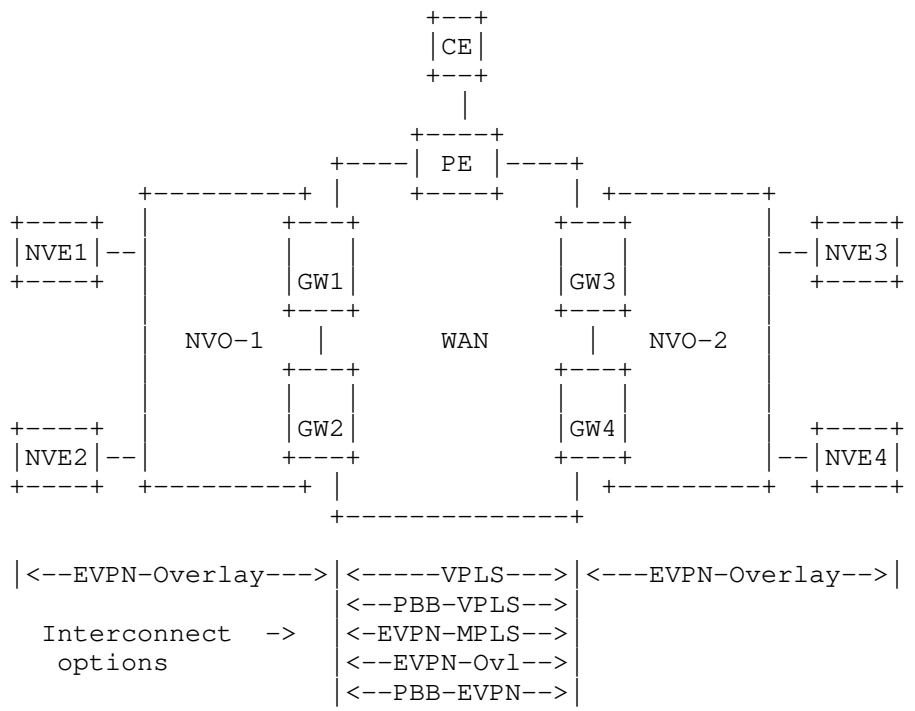


Figure 2 Integrated Interconnect model

3.1. Interconnect requirements

The solution must observe the following requirements:

- The GW function must provide control plane and data plane interworking between the EVPN-overlay network and the L2VPN technology supported in the WAN, i.e. (PBB-)VPLS or (PBB-)EVPN, as depicted in Figure 2.

- Multi-homing MUST be supported. Single-active multi-homing with per-service load balancing MUST be implemented. All-active multi-homing, i.e. per-flow load-balancing, MUST be implemented as long as the technology deployed in the WAN supports it.

- If EVPN is deployed in the WAN, the MAC Mobility, Static MAC protection and other procedures (e.g. proxy-arp) described in [EVPN] must be supported end-to-end.

- Any type of inclusive multicast tree MUST be independently supported in the WAN as per [EVPN], and in the DC as per [EVPN-Overlays].

3.2. VPLS Interconnect for EVPN-Overlay networks

3.2.1. Control/Data Plane setup procedures on the GWs

Regular MPLS tunnels and TLDP/BGP sessions will be setup to the WAN PEs and RRs as per [RFC4761][RFC4762][RFC6074] and overlay tunnels and EVPN will be setup as per [EVPN-Overlays]. Note that different route-targets for the DC and for the WAN are normally required. A single type-1 RD per service can be used.

In order to support multi-homing, the GWs will be provisioned with an I-ESI (see section 2.4), that will be unique per interconnection. All the [EVPN] procedures are still followed for the I-ESI, e.g. any MAC address learnt from the WAN will be advertised to the DC with the I-ESI in the ESI field.

A MAC-VRF per EVI will be created in each GW. The MAC-VRF will have two different types of tunnel bindings instantiated in two different split-horizon-groups:

- VPLS PWs will be instantiated in the "WAN split-horizon-group".
- Overlay tunnel bindings (e.g. VXLAN, NVGRE) will be instantiated in the "DC split-horizon-group".

Attachment circuits are also supported on the same MAC-VRF, but they will not be part of any of the above split-horizon-groups.

Traffic received in a given split-horizon-group will never be forwarded to a member of the same split-horizon-group.

As far as BUM flooding is concerned, a flooding list will be created with the sub-list created by the inclusive multicast routes and the sub-list created for VPLS in the WAN. BUM frames received from a local attachment circuit will be flooded to both sub-lists. BUM frames received from the DC or the WAN will be forwarded to the flooding list observing the split-horizon-group rule described above.

Note that the GWs are not allowed to have an EVPN binding and a PW to the same far-end within the same MAC-VRF in order to avoid loops and packet duplication. This is described in [EVPN-VPLS-INTEGRATION].

The optimizations procedures described in section 2.5 can also be applied to this model.

3.2.2. Multi-homing procedures on the GWs

Single-active multi-homing MUST be supported on the GWs. All-active multi-homing is not supported by VPLS.

All the single-active multi-homing procedures as described by [EVPN-Overlays] will be followed for the I-ESI.

The non-DF GW for the I-ESI will block the transmission and reception of all the bindings in the "WAN split-horizon-group" for BUM and unicast traffic.

3.3. PBB-VPLS Interconnect for EVPN-Overlay networks

3.3.1. Control/Data Plane setup procedures on the GWs

In this case, there is no impact on the procedures described in [RFC7041] for the B-component. However the I-component instances become EVI instances with EVPN-Overlay bindings and potentially local attachment circuits. M MAC-VRF instances can be multiplexed into the same B-component instance. This option provides significant savings in terms of PWs to be maintained in the WAN.

The I-ESI concept described in section 3.2.1 will also be used for the PBB-VPLS-based Interconnect.

B-component PWs and I-component EVPN-overlay bindings established to the same far-end will be compared. The following rules will be observed:

- Attempts to setup a PW between the two GWs within the B-component context will never be blocked.

- If a PW exists between two GWs for the B-component and an attempt is made to setup an EVPN binding on an I-component linked to that B-component, the EVPN binding will be kept operationally down. Note that the BGP EVPN routes will still be valid but not used.

- The EVPN binding will only be up and used as long as there is no PW to the same far-end in the corresponding B-component. The EVPN bindings in the I-components will be brought down before the PW in the B-component is brought up.

The optimizations procedures described in section 2.5 can also be applied to this Interconnect option.

3.3.2. Multi-homing procedures on the GWs

Single-active multi-homing MUST be supported on the GWs.

All the single-active multi-homing procedures as described by [EVPN-Overlays] will be followed for the I-ESI for each EVI instance connected to B-component.

3.4. EVPN-MPLS Interconnect for EVPN-Overlay networks

If EVPN for MPLS tunnels, EVPN-MPLS hereafter, is supported in the WAN, an end-to-end EVPN solution can be deployed. The following sections describe the proposed solution as well as the impact required on the [EVPN] procedures.

3.4.1. Control Plane setup procedures on the GWs

The GWs MUST establish separate BGP sessions for sending/receiving EVPN routes to/from the DC and to/from the WAN. Normally each GW will setup one (two) BGP EVPN session(s) to the DC RR(s) and one(two) session(s) to the WAN RR(s). The same route-distinguisher (RD) per MAC-VRF can be used for the EVPN routes sent to both, WAN and DC RRs. On the contrary, although reusing the same value is possible, different route-targets are expected to be handled for the same EVI in the WAN and the DC.

As in the other discussed options, an I-ESI will be configured on the GWs for multi-homing.

Received EVPN routes will never be reflected on the GWs but consumed and re-advertised (if needed):

- Ethernet A-D routes, ES routes and inclusive multicast routes are consumed by the GWs and processed locally for the corresponding [EVPN] procedures.

- MAC/IP advertisement routes will be received, imported and if they become active in the MAC-VRF MAC FIB, the information will be re-advertised as new routes with the following fields:

  + The RD will be the GW's RD for the MAC-VRF.

  + The ESI will be set to the I-ESI.

  + The Ethernet-tag will be 0 or a new value.

  + The MAC length, MAC address, IP Length and IP address values will be kept from the received DC NLRI.

  + The MPLS label will be a local value (when sent to the WAN) or a DC-global value (when sent to the DC).

+ The appropriate Route-Targets (RTs) and [RFC5512] BGP Encapsulation extended community will be used according to [EVPN-Overlays].

The GWs will also generate the following local EVPN routes that will be sent to the DC and WAN, with their corresponding RTs and [RFC5512] BGP Encapsulation extended community values:

- o ES route for the I-ESI.
- o Ethernet A-D routes per ESI and EVI for the I-ESI.
- o Inclusive multicast routes with independent tunnel type value for the WAN and DC. E.g. a P2MP LSP may be used in the WAN whereas ingress replication is used in the DC.
- o MAC/IP advertisement routes for MAC addresses learnt in local attachment circuits. Note that these routes will not include the I-ESI, but ESI=0 or different from 0 for local Ethernet Segments (ES).

Note that each GW will receive two copies of each of the above routes generated by the peer GW (one copy for the DC encapsulation and one copy for the WAN encapsulation). This is the expected behavior on the GW:

- o ES and A-D (per ESI) routes: regular BGP selection will be applied.
- o Inclusive multicast routes: if the Ethernet Tag ID matches on both routes, regular BGP selection applies and only one route will be active. It is recommended to influence the BGP selection so that the DC route is preferred. If the Ethernet Tag ID does not match, then BGP will consider them two separate routes. In that case, the MAC-VRF will select the DC route.
- o MAC/IP advertisement routes for local attachment circuits: as above, the GW will select only one. The decision will be made at BGP or MAC-RVRF level, depending on the Ethernet Tags.

The optimizations procedures described in section 2.5 can also be applied to this option.

3.4.2. Data Plane setup procedures on the GWs

The procedure explained at the end of the previous section will make sure there are no loops or packet duplication between the GWs of the same DC since only one EVPN binding will be setup in the data plane

between the two nodes.

As for the rest of the EVPN tunnel bindings, two flooding lists will be setup by each GW for the same MAC-VRF:

- o EVPN-overlay flooding list (composed of bindings to the remote NVEs or multicast tunnel to the NVEs).

- o EVPN-MPLS flooding list (composed of MP2P and or LSM tunnel to the remote PEs)

Each flooding list will be part of a separate split-horizon-group. Traffic generated from a local AC can be flooded to both split-horizon-groups. Traffic from a binding of a split-horizon-group can be flooded to the other split-horizon-group and local ACs, but never to a member of its own split-horizon-group.

3.4.3. Multi-homing procedures on the GWs

Single-active as well as all-active multi-homing MUST be supported.

All the multi-homing procedures as described by [EVPN] will be followed for the DF election for I-ESI, as well as the backup-path (single-active) and aliasing (all-active) procedures on the remote PEs/NVEs. The following changes are required at the GW with respect to the I-ESI:

- o Single-active multi-homing; assuming a WAN split-horizon-group, a DC split-horizon-group and local ACs on the GWs:

  + Forwarding behavior on the non-DF: the non-DF MUST NOT forward BUM or unicast traffic received from a given split-horizon-group to a member of his own split-horizon group or to the other split-horizon-group. Only forwarding to local ACs is allowed (as long as they are not part of an ES for which the node is non-DF).

  + Forwarding behavior on the DF: the DF MUST NOT forward BUM or unicast traffic received from a given split-horizon-group to a member of his own split-horizon group or to the non-DF. Forwarding to the other split-horizon-group and local ACs is allowed (as long as they are not part of an ES for which the node is non-DF).

- o All-active multi-homing; assuming a WAN split-horizon-group, a DC split-horizon-group and local ACs on the GWs:

  + Forwarding behavior on the non-DF: the non-DF follows the same

behavior as the non-DF in the single-active case but only for BUM traffic. Unicast traffic received from a split-horizon-group MUST NOT be forwarded to a member of its own split-horizon-group but can be forwarded normally to the other split-horizon-group and local ACs. If a known unicast packet is identified as a "flooded" packet, the procedures for BUM traffic MUST be followed.

+ Forwarding behavior on the DF: the DF follows the same behavior as the DF in the single-active case but only for BUM traffic. Unicast traffic received from a split-horizon-group MUST NOT be forwarded to a member of its own split-horizon-group but can be forwarded normally to the other split-horizon-group and local ACs. If a known unicast packet is identified as a "flooded" packet, the procedures for BUM traffic MUST be followed.

o No ESI label is required to be signaled for I-ESI for its use by the non-DF in the data path. This is possible because the non-DF and the DF will never forward BUM traffic (coming from a split-horizon-group) to each other.

3.4.4. Impact on MAC Mobility procedures

Since the MAC/IP Advertisement routes are not reflected in the GWs but rather consumed and re-advertised if active, the MAC Mobility procedures can be constrained to each domain (DC or WAN) and resolved within each domain. In other words, if a MAC moves within the DC, the GW MUST NOT re-advertise the route to the WAN with a change in the sequence number. Only when the MAC moves from the WAN domain to the DC domain, the GW will re-advertise the MAC with a higher sequence number in the MAC Mobility extended community. In respect to the MAC Mobility procedures described in [EVPN] the MAC addresses learnt from the NVEs in the local DC or on the local ACs will be considered as local.

The sequence numbers MUST NOT be propagated between domains. The sticky bit indication in the MAC Mobility extended community MUST be propagated between domains.

3.4.5. Gateway optimizations

All the Gateway optimizations described in section 2.5 MAY be applied to the GWs when the Interconnect is based on EVPN-MPLS.

In particular, the use of the Unknown MAC route, as described in section 2.5.1, reduces the unknown flooding in the DC but also solves some transient packet duplication issues in cases of all-active

multi-homing. This is explained in the following paragraph.

Consider the diagram in Figure 2 for EVPN-MPLS Interconnect and all-active multi-homing, and the following sequence:

a) MAC Address M1 is advertised from NVE3 in EVI-1.

b) GW3 and GW4 learn M1 for EVI-1 and re-advertise M1 to the WAN with I-ESI-2 in the ESI field.

c) GW1 and GW2 learn M1 and install GW3/GW4 as next-hops following the EVPN aliasing procedures.

d) Before NVE1 learns M1, a packet arrives to NVE1 with destination M1. The packet is subsequently flooded.

e) Since both GW1 and GW2 know M1, they both forward the packet to the WAN (hence creating packet duplication), unless there is an indication in the data plane that the packet from NVE1 has been flooded. If the GWs signal the same VNI/VSID for MAC/IP advertisement and inclusive multicast routes for EVI-1, such data plane indication does not exist.

This undesired situation can be avoided by the use of the Unknown-MAC-route. If this route is used, the NVEs will prune their unknown unicast flooding list, and the non-DF GW will not received unknown packets, only the DF will. This solves the MAC duplication issue described above.

3.4.6. Benefits of the EVPN-MPLS Interconnect solution

Besides retaining the EVPN attributes between Data Centers and throughout the WAN, the EVPN-MPLS Interconnect solution on the GWs has some benefits compared to pure BGP EVPN RR or Inter-AS model B solutions without a gateway:

- The solution supports the connectivity of local attachment circuits on the GWs.
- Different data plane encapsulations can be supported in the DC and the WAN.
- Optimized multicast solution, with independent inclusive multicast trees in DC and WAN.
- MPLS Label aggregation: for the case where MPLS labels are signaled from the NVEs for MAC/IP Advertisement routes, this solution provides label aggregation. A remote PE MAY receive a

single label per GW MAC-VRF as opposed to a label per NVE/MAC-VRF connected to the GW MAC-VRF. For instance, in Figure 2, PE would receive only one label for all the routes advertised for a given MAC-VRF from GW1, as opposed to a label per NVE/MAC-VRF.

- The GW will not propagate MAC mobility for the MACs moving within a DC. Mobility intra-DC is solved by all the NVEs in the DC. The MAC Mobility procedures on the GWs are only required in case of mobility across DCs.

- Proxy-ARP/ND function on the DGWs can be leveraged to reduce ARP/ND flooding in the DC or/and in the WAN.

3.5. PBB-EVPN Interconnect for EVPN-Overlay networks

[PBB-EVPN] is yet another Interconnect option. It requires the use of GWs where I-components and associated B-components are EVI instances.

3.5.1. Control/Data Plane setup procedures on the GWs

EVPN will run independently in both components, the I-component MAC-VRF and B-component MAC-VRF. Compared to [PBB-EVPN], the DC C-MACs are no longer learnt in the data plane on the GW but in the control plane through EVPN running on the I-component. Remote C-MACs coming from remote PEs are still learnt in the data plane. B-MACs in the B-component will be assigned and advertised following the procedures described in [PBB-EVPN].

An I-ESI will be configured on the GWs for multi-homing, but it will only be used in the EVPN control plane for the I-component EVI. No non-reserved ESIs will be used in the control plane of the B-component EVI as per [PBB-EVPN].

The rest of the control plane procedures will follow [EVPN] for the I-component EVI and [PBB-EVPN] for the B-component EVI.

From the data plane perspective, the I-component and B-component EVPN bindings established to the same far-end will be compared and the I-component EVPN-overlay binding will be kept down following the rules described in section 3.3.1.

3.5.2. Multi-homing procedures on the GWs

Single-active as well as all-active multi-homing MUST be supported.

The forwarding behavior of the DF and non-DF will be changed based on the description outlined in section 3.4.3, only replacing the "WAN

split-horizon-group" for the B-component.

3.5.3. Impact on MAC Mobility procedures

C-MACs learnt from the B-component will be advertised in EVPN within the I-component EVI scope. If the C-MAC was previously known in the I-component database, EVPN would advertise the C-MAC with a higher sequence number, as per [EVPN]. From a Mobility perspective and the related procedures described in [EVPN], the C-MACs learnt from the B-component are considered local.

3.5.4. Gateway optimizations

All the considerations explained in section 3.4.5 are applicable to the PBB-EVPN Interconnect option.

3.6. EVPN-VXLAN Interconnect for EVPN-Overlay networks

If EVPN for Overlay tunnels is supported in the WAN and a GW function is required, an end-to-end EVPN solution can be deployed. This section focuses on the specific case of EVPN for VXLAN (EVPN-VXLAN hereafter) and the impact on the [EVPN] procedures.

This use-case assumes that NVEs need to use the VNIs or VSIDs as a globally unique identifiers within a data center, and a Gateway needs to be employed at the edge of the data center network to translate the VNI or VSID when crossing the network boundaries. This GW function provides VNI and tunnel IP address translation. The use-case in which local downstream assigned VNIs or VSIDs can be used (like MPLS labels) is described by [EVPN-Overlays].

While VNIs are globally significant within each DC, there are two possibilities in the Interconnect network:

a) Globally unique VNIs in the Interconnect network:
   In this case, the GWs and PEs in the Interconnect network will agree on a common VNI for a given EVI. The RT to be used in the Interconnect network can be auto-derived from the agreed Interconnect VNI. The VNI used inside each DC MAY be the same as the Interconnect VNI.

b) Downstream assigned VNIs in the Interconnect network.
   In this case, the GWs and PEs MUST use the proper RTs to import/export the EVPN routes. Note that even if the VNI is downstream assigned in the Interconnect network, and unlike option B, it only identifies the <Ethernet Tag, GW> pair and not the <Ethernet Tag, egress PE> pair. The VNI used inside each DC MAY be the same as the Interconnect VNI. GWs SHOULD

support multiple VNI spaces per EVI (one per Interconnect network they are connected to).

In both options, NVEs inside a DC only have to be aware of a single VNI space, and only GWs will handle the complexity of managing multiple VNI spaces. In addition to VNI translation above, the GWs will provide translation of the tunnel source IP for the packets generated from the NVEs, using their own IP address. GWs will use that IP address as the BGP next-hop in all the EVPN updates to the Interconnect network.

The following sections provide more details about these two options.

3.6.1. Globally unique VNIs in the Interconnect network

Considering Figure 2, if a host H1 in NVO-1 needs to communicate with a host H2 in NVO-2, and assuming that different VNIs are used in each DC for the same EVI, e.g. VNI-10 in NVO-1 and VNI-20 in NVO-2, then the VNIs must be translated to a common Interconnect VNI (e.g. VNI-100) on the GWs. Each GW is provisioned with a VNI translation mapping so that it can translate the VNI in the control plane when sending BGP EVPN route updates to the Interconnect network. In other words, GW1 and GW2 must be configured to map VNI-10 to VNI-100 in the BGP update messages for H1's MAC route. This mapping is also used to translate the VNI in the data plane in both directions, that is, VNI-10 to VNI-100 when the packet is received from NVO-1 and the reverse mapping from VNI-100 to VNI-10 when the packet is received from the remote NVO-2 network and needs to be forwarded to NVO-1.

The procedures described in section 3.4 will be followed, considering that the VNIs advertised/received by the GWs will be translated accordingly.

3.6.2. Downstream assigned VNIs in the Interconnect network

In this case, if a host H1 in NVO-1 needs to communicate with a host H2 in NVO-2, and assuming that different VNIs are used in each DC for the same EVI, e.g. VNI-10 in NVO-1 and VNI-20 in NVO-2, then the VNIs must be translated as in section 3.6.1. However, in this case, there is no need to translate to a common Interconnect VNI on the GWs. Each GW can translate the VNI received in an EVPN update to a locally assigned VNI advertised to the Interconnect network. Each GW can use a different Interconnect VNI, hence this VNI does not need to be agreed on all the GWs and PEs of the Interconnect network.

The procedures described in section 3.4 will be followed, taking the considerations above for the VNI translation.

5. Conventions and 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 [RFC2119].

AC: Attachment Circuit

BUM: it refers to the Broadcast, Unknown unicast and Multicast traffic

DF: Designated Forwarder

GW: Gateway or Data Center Gateway

DCI: Data Center Interconnect

ES: Ethernet Segment

ESI: Ethernet Segment Identifier

I-ESI: Interconnect ESI defined on the GWs for multi-homing to/from the WAN

EVI: EVPN Instance

MAC-VRF: it refers to an EVI instance in a particular node

NVE: Network Virtualization Edge

PW: Pseudowire

RD: Route-Distinguisher

RT: Route-Target

TOR: Top-Of-Rack switch

VNI/VSID: refers to VXLAN/NVGRE virtual identifiers

VSI: Virtual Switch Instance or VPLS instance in a particular PE

6. Security Considerations

This section will be completed in future versions.

7. IANA Considerations

8. References

8.1. Normative References

[RFC4761]Kompella, K., Ed., and Y. Rekhter, Ed., "Virtual Private LAN Service (VPLS) Using BGP for Auto-Discovery and Signaling", RFC 4761, 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, January 2007, <http://www.rfc-editor.org/info/rfc4762>.

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

[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, November 2013, <http://www.rfc-editor.org/info/rfc7041>.

8.2. Informative References

[EVPN] Sajassi et al., "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-10.txt, work in progress, October, 2014

[PBB-EVPN] Sajassi et al., "PBB-EVPN", draft-ietf-l2vpn-pbb-evpn-07, work in progress, June, 2014

[EVPN-Overlays] Sajassi-Drake et al., "A Network Virtualization Overlay Solution using EVPN", draft-sd-l2vpn-evpn-overlay-03.txt, work in progress, June, 2014

[EVPN-VPLS-INTEGRATION] Sajassi et al., "(PBB-)EVPN Seamless Integration with (PBB-)VPLS", draft-sajassi-l2vpn-evpn-vpls-integration-01.txt, work in progress, October, 2014

9. Acknowledgments

This document was prepared using 2-Word-v2.0.template.dot.

10. Authors' Addresses

Jorge Rabadan
Alcatel-Lucent
777 E. Middlefield Road
Mountain View, CA 94043 USA
Email: jorge.rabadan@alcatel-lucent.com

Senthil Sathappan
Alcatel-Lucent
Email: senthil.sathappan@alcatel-lucent.com

Wim Henderickx
Alcatel-Lucent
Email: wim.henderickx@alcatel-lucent.com

Florin Balus
Nuage Networks
Email: florin@nuagenetworks.net

Senad Palislamovic
Alcatel-Lucent
Email: senad.palislamovic@alcatel-lucent.com

Ali Sajassi
Cisco
Email: sajassi@cisco.com

Ravi Shekhar
Juniper
Email: rshekhar@juniper.net

Anil Lohiya
Juniper
Email: alohiya@juniper.net

Dennis Cai
Cisco Systems
Email: dcai@cisco.com

L2VPN Workgroup
Internet Draft

Intended status: Standards Track

J. Rabadan
W. Henderickx
S. Palislamovic
Alcatel-Lucent

J. Drake
Juniper

F. Balus
Nuage Networks

A. Sajassi
Cisco

A. Isaac
Bloomberg

Expires: April 19, 2015

October 16, 2014

IP Prefix Advertisement in EVPN
draft-rabadan-l2vpn-evpn-prefix-advertisement-03

Abstract

EVPN provides a flexible control plane that allows intra-subnet connectivity in an IP/MPLS and/or an NVO-based network. In NVO networks, there is also a need for a dynamic and efficient inter-subnet connectivity across Tenant Systems and End Devices that can be physical or virtual and may not support their own routing protocols. This document defines a new EVPN route type for the advertisement of IP Prefixes and explains some use-case examples where this new route-type is used.

This Internet-Draft will expire on April 19, 2015.

Table of Contents

1. Terminology

GW IP: Gateway IP Address

IPL: IP address length

IRB: Integrated Routing and Bridging interface

ML: MAC address length

NVE: Network Virtualization Edge

TS: Tenant System

VA: Virtual Appliance

RT-2: EVPN route type 2, i.e. MAC/IP advertisement route

RT-5: EVPN route type 5, i.e. IP Prefix route

Overlay next-hop: object used in the IP Prefix route, as described in this document. It can be an IP address in the tenant space or an ESI, and identifies the next-hop yielded by the IP route lookup at the routing context importing the route. An overlay next-hop always needs a recursive route resolution on the NVE receiving the IP Prefix route, so that the NVE knows to which egress NVE to forward the packets.

Underlay next-hop: IP address sent by BGP along with any EVPN route, i.e. BGP next-hop. It identifies the NVE sending the route and it is used at the receiving NVE as the VXLAN destination VTEP or NVGRE destination end-point.

2. Introduction and problem statement

Inter-subnet connectivity is required for certain tenants within the Data Center. [EVPN-INTERSUBNET] defines some fairly common inter-subnet forwarding scenarios where TSes can exchange packets with TSes located in remote subnets. In order to meet this requirement, [EVPN-INTERSUBNET] describes how MAC/IPs encoded in TS RT-2 routes are not only used to populate MAC-VRF and overlay ARP tables, but also IP-VRF tables with the encoded TS host routes (/32 or /128). In some cases, EVPN may advertise IP Prefixes and therefore provide aggregation in the IP-VRF tables, as opposed to program individual host routes. This document complements the scenarios described in [EVPN-INTERSUBNET] and defines how EVPN may be used to advertise IP Prefixes.

Section 2.1 describes the inter-subnet connectivity requirements in Data Centers. Section 2.2 explains why a new EVPN route type is required for IP Prefix advertisements. Once the need for a new EVPN route type is justified, sections 3, 4 and 5 will describe this route type and how it is used in some specific use cases.

2.1 Inter-subnet connectivity requirements in Data Centers

[EVPN] is used as the control plane for a Network Virtualization Overlay (NVO3) solution in Data Centers (DC), where Network Virtualization Edge (NVE) devices can be located in Hypervisors or TORs, as described in [EVPN-OVERLAYS].

If we use the term Tenant System (TS) to designate a physical or virtual system identified by MAC and IP addresses, and connected to an EVPN instance, the following considerations apply:

- The Tenant Systems may be Virtual Machines (VMs) that generate traffic from their own MAC and IP.
- The Tenant Systems may be Virtual Appliance entities (VAs) that forward traffic to/from IP addresses of different End Devices seating behind them.
    - These VAs can be firewalls, load balancers, NAT devices, other appliances or virtual gateways with virtual routing instances.
    - These VAs do not have their own routing protocols and hence rely on the EVPN NVEs to advertise the routes on their behalf.
    - In all these cases, the VA will forward traffic to the Data Center using its own source MAC but the source IP will be the one associated to the End Device seating behind or a translated IP address (part of a public NAT pool) if the VA is performing NAT.
    - Note that the same IP address could exist behind two of these TS. One example of this would be certain appliance resiliency mechanisms, where a virtual IP or floating IP can be owned by one of the two VAs running the resiliency protocol (the master VA). VRRP is one particular example of this. Another example is multi-homed subnets, i.e. the same subnet is connected to two VAs.
    - Although these VAs provide IP connectivity to VMs and subnets behind them, they do not always have their own IP interface connected to the EVPN NVE, e.g. layer-2 firewalls are examples of VAs not supporting IP interfaces.

The following figure illustrates some of the examples described above.

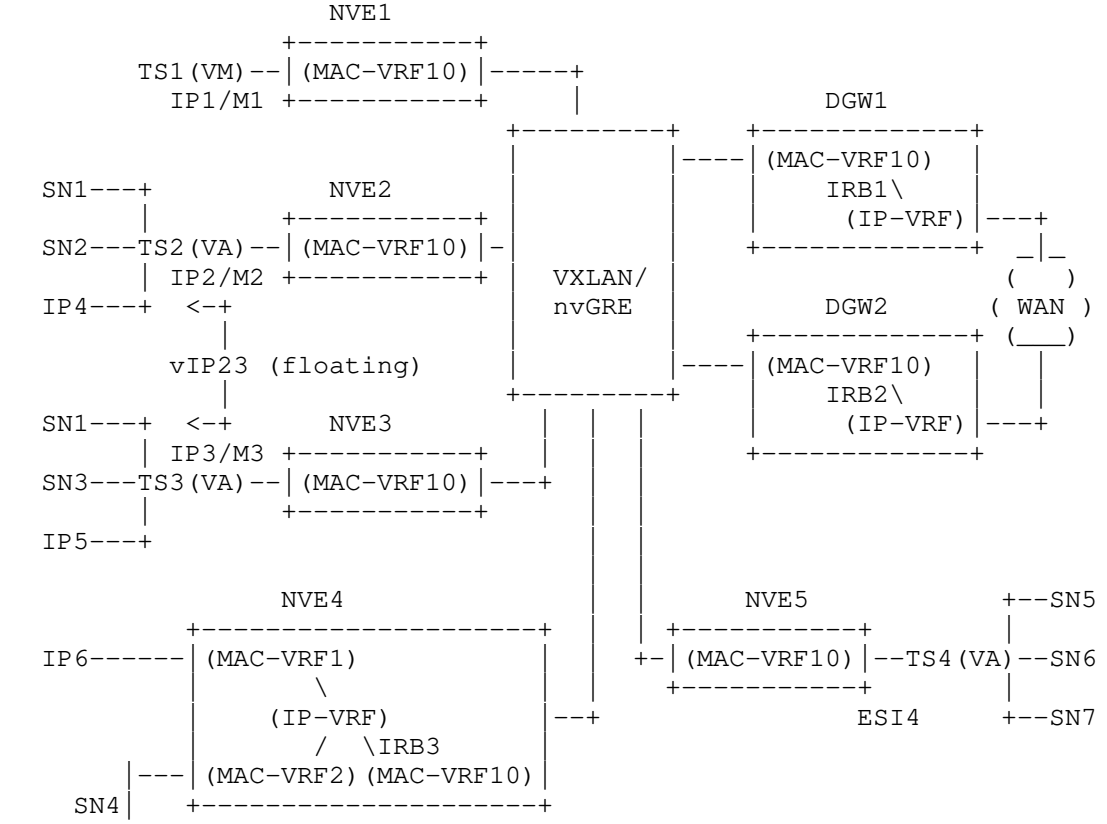


Figure 1 DC inter-subnet use-cases

Where:

NVE1, NVE2, NVE3, NVE4, NVE5, DGW1 and DGW2 share the same EVI for a particular tenant. EVI-10 is comprised of the collection of MAC-VRF10 instances defined in all the NVEs. All the hosts connected to EVI-10 belong to the same IP subnet. The hosts connected to EVI-10 are listed below:

- TS1 is a VM that generates/receives traffic from/to IP1, where IP1 belongs to the EVI-10 subnet.

- TS2 and TS3 are Virtual Appliances (VA) that generate/receive traffic from/to the subnets and hosts seating behind them (SN1, SN2, SN3, IP4 and IP5). Their IP addresses (IP2 and IP3) belong to the EVI-10 subnet and they can also generate/receive

traffic. When these VAs receive packets destined to their own MAC addresses (M2 and M3) they will route the packets to the proper subnet or host. These VAs do not support routing protocols to advertise the subnets connected to them and can move to a different server and NVE when the Cloud Management System decides to do so. These VAs may also support redundancy mechanisms for some subnets, similar to VRRP, where a floating IP is owned by the master VA and only the master VA forwards traffic to a given subnet. E.g.: vIP23 in figure 1 is a floating IP that can be owned by TS2 or TS3 depending on who the master is. Only the master will forward traffic to SN1.

- o Integrated Routing and Bridging interfaces IRB1, IRB2 and IRB3 have their own IP addresses that belong to the EVI-10 subnet too. These IRB interfaces connect the EVI-10 subnet to Virtual Routing and Forwarding (VRF) instances that can route the traffic to other connected subnets for the same tenant (within the DC or at the other end of the WAN).

- o TS4 is a layer-2 VA that provides connectivity to subnets SN5, SN6 and SN7, but does not have an IP address itself in the EVI-10. TS4 is connected to a physical port on NVE5 assigned to Ethernet Segment Identifier 4.

All the above DC use cases require inter-subnet forwarding and therefore the individual host routes and subnets:

a) MUST be advertised from the NVEs (since VAs and VMs do not run routing protocols) and
b) MAY be associated to an overlay next-hop that can be a VA IP address, a floating IP address or an ESI.

2.2 The requirement for a new EVPN route type

[EVPN] defines a MAC/IP route (also referred as RT-2) where a MAC address can be advertised together with an IP address length (IPL) and IP address (IP). While a variable IPL might have been used to indicate the presence of an IP prefix in a route type 2, there are several specific use cases in which using this route type to deliver IP Prefixes is not suitable.

One example of such use cases is the "floating IP" example described in section 2.1. In this example we need to decouple the advertisement of the prefixes from the advertisement of the floating IP (vIP23 in figure 1) and MAC associated to it, otherwise the solution gets highly inefficient and does not scale.

E.g.: if we are advertising 1k prefixes from M2 (using RT-2) and the

floating IP owner changes from M2 to M3, we would need to withdraw 1k routes from M2 and re-advertise 1k routes from M3. However if we use a separate route type, we can advertise the 1k routes associated to the floating IP address (vIP23) and only one RT-2 for advertising the ownership of the floating IP, i.e. vIP23 and M2 in the route type 2. When the floating IP owner changes from M2 to M3, a single RT-2 withdraw/update is required to indicate the change. The remote DGW will not change any of the 1k prefixes associated to vIP23, but will only update the ARP resolution entry for vIP23 (now pointing at M3).

Other reasons to decouple the IP Prefix advertisement from the MAC/IP route are listed below:

- Clean identification, operation of troubleshooting of IP Prefixes, not subject to interpretation and independent of the IPL and the IP value. E.g.: a default IP route 0.0.0.0/0 must always be easily and clearly distinguished from the absence of IP information.

- MAC address information must not be compared by BGP when selecting two IP Prefix routes. If IP Prefixes were to be advertised using MAC/IP routes, the MAC information would always be present and part of the route key.

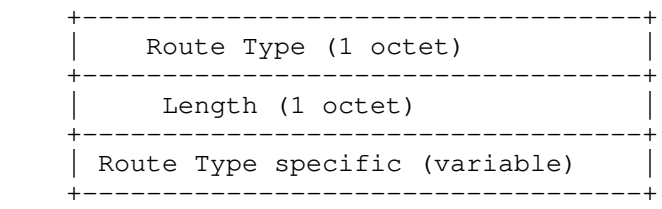
- IP Prefix routes must not be subject to MAC/IP route procedures such as MAC mobility or aliasing. Prefixes advertised from two different ESIs do not mean mobility; MACs advertised from two different ESIs do mean mobility. Similarly load balancing for IP prefixes is achieved through IP mechanisms such as ECMP, and not through MAC route mechanisms such as aliasing.

- NVEs that do not require processing IP Prefixes must have an easy way to identify an update with an IP Prefix and ignore it, rather than processing the MAC/IP route to find out only later that it carries a Prefix that must be ignored.

The following sections describe how EVPN is extended with a new route type for the advertisement of IP prefixes and how this route is used to address the current and future inter-subnet connectivity requirements existing in the Data Center.

3. The BGP EVPN IP Prefix route

The current BGP EVPN NLRI as defined in [EVPN] is shown below:



Where the route type field can contain one of the following specific values:

+ 1 - Ethernet Auto-Discovery (A-D) route

+ 2 - MAC/IP advertisement route

+ 3 - Inclusive Multicast Route

+ 4 - Ethernet Segment Route

This document defines an additional route type that will be used for the advertisement of IP Prefixes:

+ 5 - IP Prefix Route

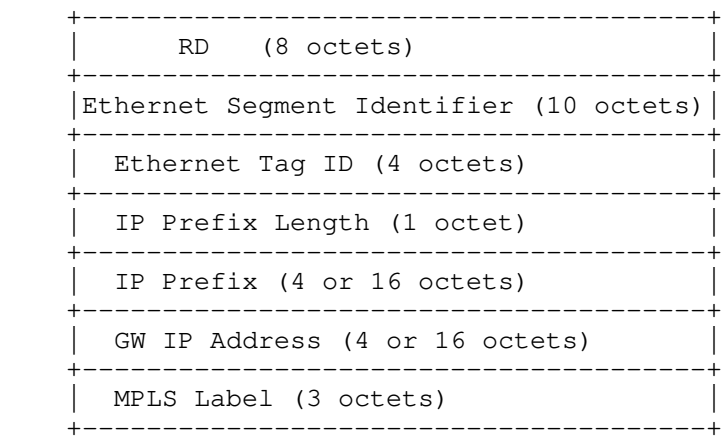
The support for this new route type is OPTIONAL.

Since this new route type is OPTIONAL, an implementation not supporting it MUST ignore the route, based on the unknown route type value.

The detailed encoding of this route and associated procedures are described in the following sections.

3.1 IP Prefix Route encoding

An IP Prefix advertisement route NLRI consists of the following fields:



Where:

- RD, Ethernet Tag ID and MPLS Label fields will be used as defined in [EVPN] and [EVPN-OVERLAYS].
- The Ethernet Segment Identifier will be a non-zero 10-byte identifier if the ESI is used as an overlay next-hop. It will be zero otherwise.
- The IP Prefix Length can be set to a value between 0 and 32 (bits) for ipv4 and between 0 and 128 for ipv6.
- The IP Prefix will be a 32 or 128-bit field (ipv4 or ipv6).
- The GW IP (Gateway IP Address) will be a 32 or 128-bit field (ipv4 or ipv6), and will encode the overlay IP next-hop for the IP Prefixes. The GW IP field can be zero if it is not used as an overlay next-hop.
- The total route length will indicate the type of prefix (ipv4 or ipv6) and the type of GW IP address (ipv4 or ipv6). Note that the IP Prefix + the GW IP should have a length of either 64 or 256 bits, but never 160 bits (ipv4 and ipv6 mixed values are not allowed).

The Eth-Tag ID, IP Prefix Length and IP Prefix will be part of the route key used by BGP to compare routes. The rest of the fields will not be part of the route key.

The route will contain a single overlay next-hop at most, i.e. if the ESI field is different from zero, the GW IP field will be zero, and vice versa. The following table shows the different inter-subnet use-

cases described in this document and the corresponding coding of the overlay next-hop in the route type 5 (RT-5). The IP-VRF-to-IP-VRF or IRB forwarding on NVEs case is a special use-case, where there is no need for overlay next-hop, since the actual next-hop is given by the BGP next-hop. When an overlay next-hop is present in the RT-5, the receiving NVE will need to perform a recursive route resolution to find out to which egress NVE to forward the packets.

Use-case	Next-hop in the RT-5 BGP update
TS IP address	GW IP Address
Floating IP address	GW IP Address
"Bump in the wire"	ESI
IP-VRF-to-IP-VRF	BGP next-hop

4. Benefits of using the EVPN IP Prefix route

This section clarifies the different functions accomplished by the EVPN RT-2 and RT-5 routes, and provides a list of benefits derived from using a separate route type for the advertisement of IP Prefixes in EVPN.

[EVPN] describes the content of the BGP EVPN RT-2 specific NLRI, i.e. MAC/IP Advertisement Route, where the IP address length (IPL) and IP address (IP) of a specific advertised MAC are encoded. The subject of the MAC advertisement route is the MAC address (M) and MAC address length (ML) encoded in the route. The MAC mobility and other complex procedures are defined around that MAC address. The IP address information carries the host IP address required for the ARP resolution of the MAC according to [EVPN] and the host route to be programmed in the IP-VRF [EVPN-INTERSUBNET].

The BGP EVPN route type 5 defined in this document, i.e. IP Prefix Advertisement route, decouples the advertisement of IP prefixes from the advertisement of any MAC address related to it. This brings some major benefits to NVO-based networks where certain inter-subnet forwarding scenarios are required. Some of those benefits are:

a) Upon receiving a route type 2 or type 5, an egress NVE can easily distinguish MACs and IPs from IP Prefixes. E.g. an IP prefix with IPL=32 being advertised from two different ingress NVEs (as RT-5) can be identified as such and be imported in the designated routing context as two ECMP routes, as opposed to two MACs competing for the same IP.

b) Similarly, upon receiving a route, an ingress NVE not supporting

processing of IP Prefixes can easily ignore the update, based on the route type.

c) A MAC route includes the ML, M, IPL and IP in the route key that is used by BGP to compare routes, whereas for IP Prefix routes, only IPL and IP (as well as Ethernet Tag ID) are part of the route key. Advertised IP Prefixes are imported into the designated routing context, where there is no MAC information associated to IP routes. In the example illustrated in figure 1, subnet SN1 should be advertised by NVE2 and NVE3 and interpreted by DGW1 as the same route coming from two different next-hops, regardless of the MAC address associated to TS2 or TS3. This is easily accomplished in the RT-5 by including only the IP information in the route key.

d) By decoupling the MAC from the IP Prefix advertisement procedures, we can leave the IP Prefix advertisements out of the MAC mobility procedures defined in [EVPN] for MACs. In addition, this allows us to have an indirection mechanism for IP Prefixes advertised from a MAC/IP that can move between hypervisors. E.g. if there are 1,000 prefixes seating behind TS2 (figure 1), NVE2 will advertise all those prefixes in RT-5 routes associated to the next-hop IP2. Should TS2 move to a different NVE, a single MAC advertisement route withdraw for the M2/IP2 route from NVE2 will invalidate the 1,000 prefixes, as opposed to have to wait for each individual prefix to be withdrawn. This may be easily accomplished by using IP Prefix routes that are not tied to a MAC address, and use a different MAC/IP route to advertise the location and resolution of the overlay next-hop to a MAC address.

5. IP Prefix next-hop use-cases

The IP Prefix route can use a GW IP or an ESI as an overlay next-hop as well as no overlay next-hop whatsoever. This section describes some use-cases for these next-hop types.

5.1 TS IP address next-hop use-case

The following figure illustrates an example of inter-subnet forwarding for subnets seating behind Virtual Appliances (on TS2 and TS3).

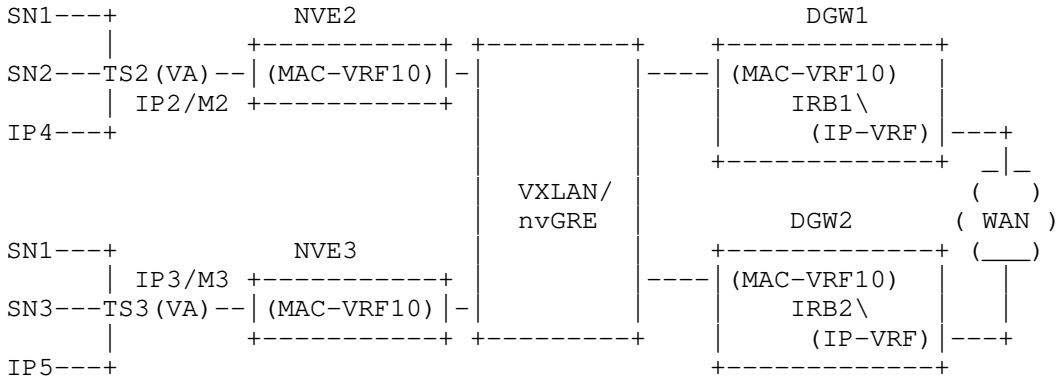


Figure 2 TS IP address use-case

An example of inter-subnet forwarding between subnet SN1/24 and a subnet seating in the WAN is described below. NVE2, NVE3, DGW1 and DGW2 are running BGP EVPN. TS2 and TS3 do not support routing protocols, only a static route to forward the traffic to the WAN.

(1) NVE2 advertises the following BGP routes on behalf of TS2:

- Route type 2 (MAC/IP route) containing: ML=48, M=M2, IPL=32, IP=IP2 and [RFC5512] BGP Encapsulation Extended Community with Tunnel-type= VXLAN or NVGRE.

- Route type 5 (IP Prefix route) containing: IPL=24, IP=SN1, ESI=0, GW IP address=IP2 (and BGP Encapsulation Extended Community).

(2) NVE3 advertises the following BGP routes on behalf of TS3:

- Route type 2 (MAC/IP route) containing: ML=48, M=M3, IPL=32, IP=IP3 (and BGP Encapsulation Extended Community).

- Route type 5 (IP Prefix route) containing: IPL=24, IP=SN1, ESI=0, GW IP address=IP3 (and BGP Encapsulation Extended Community).

(3) DGW1 and DGW2 import both received routes based on the route-targets:

- Based on the MAC-VRF10 route-target in DGW1 and DGW2, the MAC/IP route is imported and M2 is added to the MAC-VRF10 along with its corresponding tunnel information. For instance, if VXLAN is used, the VTEP will be derived from the MAC/IP route BGP next-hop (underlay next-hop) and VNI from the

Ethernet Tag or MPLS fields. IP2 - M2 is added to the ARP table.

- Based on the MAC-VRF10 route-target in DGW1 and DGW2, the IP Prefix route is also imported and SN1/24 is added to the designated routing context with next-hop IP2 pointing at the local MAC-VRF10. Should ECMP be enabled in the routing context, SN1/24 would also be added to the routing table with next-hop IP3.

(4) When DGW1 receives a packet from the WAN with destination IPx, where IPx belongs to SN1/24:

- A destination IP lookup is performed on the DGW1 IP-VRF routing table and next-hop=IP2 is found. Since IP2 is an overlay next-hop a recursive route resolution is required for IP2.

- IP2 is resolved to M2 in the ARP table, and M2 is resolved to the tunnel information given by the MAC FIB (remote VTEP and VNI for the VXLAN case).

- The IP packet destined to IPx is encapsulated with:

  . Source inner MAC = IRB1 MAC

  . Destination inner MAC = M2

  . Tunnel information provided by the MAC-VRF (VNI, VTEP IPs and MACs for the VXLAN case)

(5) When the packet arrives at NVE2:

- Based on the tunnel information (VNI for the VXLAN case), the MAC-VRF10 context is identified for a MAC lookup.

- Encapsulation is stripped-off and based on a MAC lookup (assuming MAC forwarding on the egress NVE), the packet is forwarded to TS2, where it will be properly routed.

(6) Should TS2 move from NVE2 to NVE3, MAC Mobility procedures will be applied to the MAC route IP2/M2, as defined in [EVPN]. Route type 5 prefixes are not subject to MAC mobility procedures, hence no changes in the DGW VRF routing table will occur for TS2 mobility, i.e. all the prefixes will still be pointing at IP2 as next-hop. There is an indirection for e.g. SN1/24, which still points at next-hop IP2 in the routing table, but IP2 will be simply resolved to a different tunnel, based on the outcome of the MAC mobility

procedures for the MAC/IP route IP2/M2.

Note that in the opposite direction, TS2 will send traffic based on its static-route next-hop information (IRB1 and/or IRB2), and regular EVPN procedures will be applied.

5.2 Floating IP next-hop use-case

Sometimes Tenant Systems (TS) work in active/standby mode where an upstream floating IP - owned by the active TS - is used as the next-hop to get to some subnets behind. This redundancy mode, already introduced in section 2.1 and 2.2, is illustrated in Figure 3.

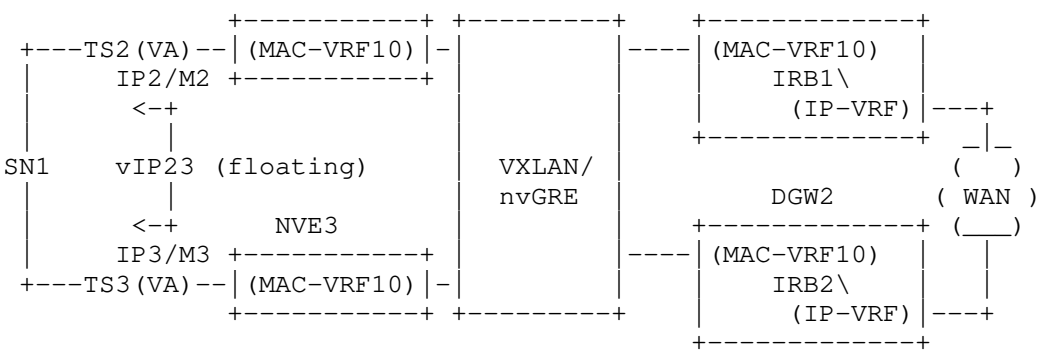


Figure 3 Floating IP next-hop for redundant TS

In this example, assuming TS2 is the active TS and owns IP23:

(1) NVE2 advertises the following BGP routes for TS2:

- Route type 2 (MAC/IP route) containing: ML=48, M=M2, IPL=32, IP=IP23 (and BGP Encapsulation Extended Community).
- Route type 5 (IP Prefix route) containing: IPL=24, IP=SN1, ESI=0, GW IP address=IP23 (and BGP Encapsulation Extended Community).

(2) NVE3 advertises the following BGP routes for TS3:

- Route type 5 (IP Prefix route) containing: IPL=24, IP=SN1, ESI=0, GW IP address=IP23 (and BGP Encapsulation Extended Community).

(3) DGW1 and DGW2 import both received routes based on the route-target:

- o M2 is added to the MAC-VRF10 MAC FIB along with its corresponding tunnel information. For the VXLAN use case, the VTEP will be derived from the MAC/IP route BGP next-hop and VNI from the Ethernet Tag or MPLS fields. IP23 - M2 is added to the ARP table.

- o SN1/24 is added to the designated routing context in DGW1 and DGW2 with next-hop IP23 pointing at the local MAC-VRF10.

(4) When DGW1 receives a packet from the WAN with destination IPx, where IPx belongs to SN1/24:

- o A destination IP lookup is performed on the DGW1 IP-VRF routing table and next-hop=IP23 is found. Since IP23 is an overlay next-hop, a recursive route resolution for IP23 is required.

- o IP23 is resolved to M2 in the ARP table, and M2 is resolved to the tunnel information given by the MAC-VRF (remote VTEP and VNI for the VXLAN case).

- o The IP packet destined to IPx is encapsulated with:

  . Source inner MAC = IRB1 MAC

  . Destination inner MAC = M2

  . Tunnel information provided by the MAC FIB (VNI, VTEP IPs and MACs for the VXLAN case)

(5) When the packet arrives at NVE2:

- o Based on the tunnel information (VNI for the VXLAN case), the MAC-VRF10 context is identified for a MAC lookup.

- o Encapsulation is stripped-off and based on a MAC lookup (assuming MAC forwarding on the egress NVE), the packet is forwarded to TS2, where it will be properly routed.

(6) When the redundancy protocol running between TS2 and TS3 appoints TS3 as the new active TS for SN1, TS3 will now own the floating IP23 and will signal this new ownership (GARP message or similar). Upon receiving the new owner's notification, NVE3 will issue a route type 2 for M3-IP23. DGW1 and DGW2 will update their ARP tables with the new MAC resolving the floating IP. No changes are carried out in the VRF routing table.

In the DGW1/2 BGP RIB, there will be two route type 5 routes for SN1

(from NVE2 and NVE3) but only the one with the same BGP next-hop as the IP23 RT-2 BGP next-hop will be valid.

5.3 ESI next-hop ("Bump in the wire") use-case

The following figure illustrates and example of inter-subnet forwarding for a subnet route that uses an ESI as an overlay next-hop. In this use-case, TS2 and TS3 are layer-2 VA devices without any IP address that can be included as an overlay next-hop in the GW IP field of the IP Prefix route.

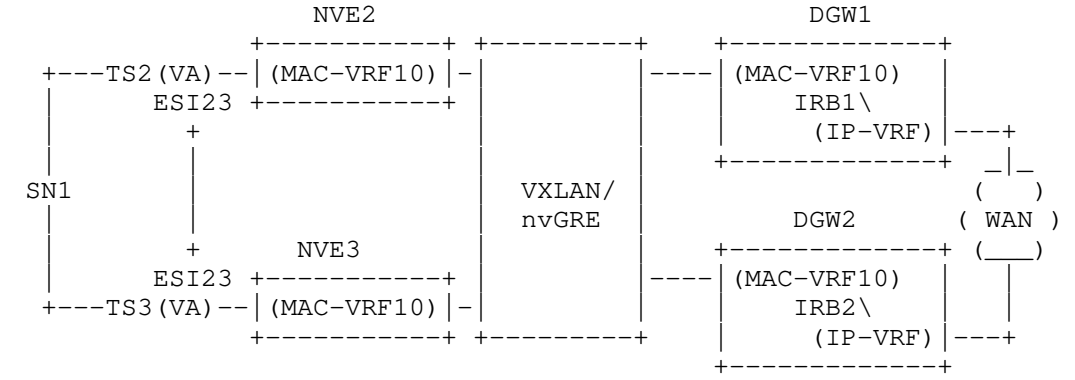


Figure 5 ESI next-hop use-case

Since neither TS2 nor TS3 can run any routing protocol and have no IP address assigned, an ESI, i.e. ESI23, will be provisioned on the attachment ports of NVE2 and NVE3. This model supports VA redundancy in a similar way as the one described in section 5.2 for the floating IP next-hop use-case, only using the EVPN Ethernet A-D route instead of the MAC advertisement route to advertise the location of the overlay next-hop. The procedure is explained below:

(1) NVE2 advertises the following BGP routes for TS2:

- Route type 1 (Ethernet A-D route for EVI-10) containing: ESI=ESI23 and the corresponding tunnel information (Ethernet Tag and/or MPLS label), as well as the BGP Encapsulation Extended Community. Assuming the ESI is active on NVE2, NVE2 will advertise this route.

- Route type 5 (IP Prefix route) containing: IPL=24, IP=SN1, ESI=ESI23, GW IP address=0 (and BGP Encapsulation Extended Community).

(2) NVE3 advertises the following BGP routes for TS3:

- Route type 1 (Ethernet A-D route for EVI-10) containing: ESI=ESI23 and the corresponding tunnel information (Ethernet Tag and/or MPLS label), as well as the BGP Encapsulation Extended Community. NVE3 will advertise this route assuming the ESI is active on NVE2. Note that if the resiliency mechanism for TS2 and TS3 is in active-active mode, both NVE2 and NVE3 will send the A-D route. Otherwise, that is, the resiliency is active-standby, only the NVE owning the active ESI will advertise the Ethernet A-D route for ESI23.

- Route type 5 (IP Prefix route) containing: IPL=24, IP=SN1, ESI=23, GW IP address=0 (and BGP Encapsulation Extended Community).

(3) DGW1 and DGW2 import the received routes based on the route-target:

- The tunnel information to get to ESI23 is installed in DGW1 and DGW2. For the VXLAN use case, the VTEP will be derived from the Ethernet A-D route BGP next-hop and VNI from the Ethernet Tag or MPLS fields (see [EVPN-OVERLAYS]).

- SN1/24 is added to the designated routing context in DGW1 and DGW2 with next-hop ESI23 pointing at the local MAC-VRF10.

(4) When DGW1 receives a packet from the WAN with destination IPx, where IPx belongs to SN1/24:

- A destination IP lookup is performed on the DGW1 IP-VRF routing table and next-hop=ESI23 is found. Since ESI23 is an overlay next-hop, a recursive route resolution is required to find the egress NVE where ESI23 resides.

- The IP packet destined to IPx is encapsulated with:

  . Source inner MAC = IRB1 MAC

  . Destination inner MAC = M2 (this MAC will be obtained after a lookup in the IP-VRF ARP table or in the MAC-VRF10 FDB table associated to ESI23).

  . Tunnel information provided by the Ethernet A-D route for ESI23 (VNI, VTEP IP and MACs for the VXLAN case).

(5) When the packet arrives at NVE2:

- Based on the tunnel information (VNI for the VXLAN case), the MAC-VRF10 context is identified for a MAC lookup (assuming MAC disposition model).

- Encapsulation is stripped-off and based on a MAC lookup (assuming MAC forwarding on the egress NVE), the packet is forwarded to TS2, where it will be properly forwarded.

(6) If the redundancy protocol running between TS2 and TS3 follows an active/standby model and there is a failure, appointing TS3 as the new active TS for SN1, TS3 will now own the connectivity to SN1 and will signal this new ownership. Upon receiving the new owner's notification, NVE3 will issue a route type 1 for ESI23, whereas NVE2 will withdraw its Ethernet A-D route for ESI23. DGW1 and DGW2 will update their tunnel information to resolve ESI23. No changes are carried out in the IP-VRF routing table.

In the DGW1/2 BGP RIB, there will be two route type 5 routes for SN1 (from NVE2 and NVE3) but only the one with the same BGP next-hop as the ESI23 route type 1 BGP next-hop will be valid.

5.4 IRB forwarding on NVEs for Subnets (IP-VRF-to-IP-VRF)

This use-case is similar to the scenario described in "IRB forwarding on NVEs for Tenant Systems" in [EVPN-INTERSUBNET], however the new requirement here is the advertisement of IP Prefixes as opposed to only host routes. In the previous examples, the MAC-VRF instance can connect IRB interfaces and any other Tenant Systems connected to it. EVPN provides connectivity for:

a) Traffic destined to the IRB IP interfaces as well as

b) Traffic destined to IP subnets seating behind the TS, e.g. SN1 or SN2.

In order to provide connectivity for (a) we need MAC/IP routes (RT-2) distributing IRB MACs and IPs. Connectivity type (b) is accomplished by the exchange of IP Prefix routes (RT-5) for IPs and subnets seating behind certain overlay next-hops.

In some cases, subnets may be advertised in IP Prefix routes without any overlay next-hop since the RT-5 itself provides all the forwarding information required to send the packets to the egress NVE and no recursive route resolution is needed. This use case is depicted in the diagram below and we refer to it as the "IRB forwarding on NVEs for Subnets" or "IP-VRF-to-IP-VRF" use-case:

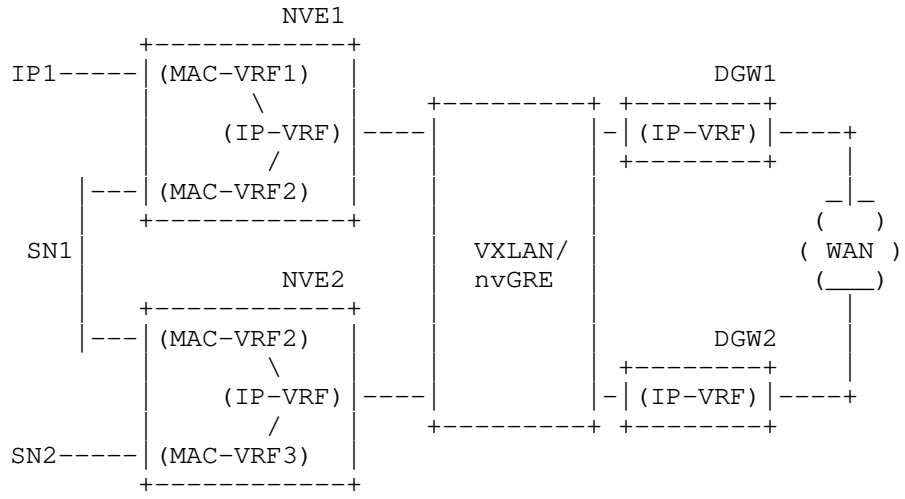


Figure 6 Inter-subnet forwarding on NVEs for Subnets

In this case, we need to provide connectivity from/to IP hosts in SN1, SN2, IP1 and hosts seating at the other end of the WAN. There is no need to define IRB interfaces to interconnect the IP-VRF instances among the NVEs for the same tenant. This is the reason why we refer to this solution as "IP-VRF-to-IP-VRF" solution.

In this case, the EVPN route type 5 will be used to advertise the IP Prefixes, along with the Router's MAC Extended Community as defined in [EVPN-INTERSUBNET]. Each NVE/DGW will advertise an RT-5 for each of its subnet prefixes with the following fields:

- o RD as per [EVPN].
- o Eth-Tag ID = 0 assuming VLAN-based service.
- o IP address length and IP address, as explained in the previous sections.
- o GW IP address=0 and ESI=0, that is, no overlay next-hop is required in this use-case, since the BGP next-hop is enough to find the egress NVE to forward the packets to.
- o MPLS label or VNID corresponding to the IP-VRF.

Each RT-5 will be sent with a route-target identifying the tenant (IP-VRF) and two BGP extended communities:

- The first one is the BGP Encapsulation Extended Community, as per [RFC5512], identifying the tunnel type.

- The second one is the Router's MAC Extended Community as per [EVPN-INTERSUBNET] containing the MAC address associated to the NVE advertising the route. This MAC address identifies the NVE/DGW and MAY be re-used for all the IP-VRFs in the node. The ingress NVE will use this MAC address as the inner MAC destination address in the packets forwarded to the owner of the RT-5.

Example of prefix advertisement for the ipv4 prefix SN1/24 advertised from NVE1:

(1) NVE1 advertises the following BGP route for SN1:

- Route type 5 (IP Prefix route) containing: Eth-Tag=0, IPL=24, IP=SN1, MPLS Label=10. An [RFC5512] BGP Encapsulation Extended Community will be sent, where Tunnel-type= VXLAN or NVGRE. A Router's MAC Extended Community will also be sent along with the RT-5, where the Router's MAC address value will contain the NVE1 MAC.

(2) DGW1 imports the received route from NVE1 and SN1/24 is added to the designated IP-VRF. The next-hop for SN1/24 will be given by the route type 5 BGP next-hop (NVE1), which is resolved to a tunnel. For instance: if the tunnel is VXLAN based, the BGP next-hop will be resolved to a VXLAN tunnel where: destination-VTEP= NVE1 IP, VNI=10, inner destination MAC = NVE1 MAC (derived from the Router's MAC Extended Community value).

(3) When DGW1 receives a packet from the WAN with destination IPx, where IPx belongs to SN1/24:

- A destination IP lookup is performed on the DGW1 IP-VRF routing table and next-hop= "NVE1 IP" is found. The tunnel information to encapsulate the packet will be derived from the route type 5 received for SN1.

- The IP packet destined to IPx is encapsulated with: Source inner MAC = DGW1 MAC, Destination inner MAC = NVE1 MAC, Source outer IP (source VTEP) = DGW1 IP, Destination outer IP (destination VTEP) = NVE1 IP.

(4) When the packet arrives at NVE1:

- Based on the tunnel information (VNI for the VXLAN case), the routing context is identified for an IP lookup.

- An IP lookup is performed in the routing context, where SN1 turns out to be a local subnet associated to MAC-VRF2. A subsequent lookup in the ARP table and the MAC-VRF FIB will return the forwarding information for the packet in EVI-2.

6. Conclusions

A new EVPN route type 5 for the advertisement of IP Prefixes is described in this document. This new route type has a differentiated role from the RT-2 route and addresses all the Data Center (or NVO-based networks in general) inter-subnet connectivity scenarios in which an IP Prefix advertisement is required. Using this new RT-5, an IP Prefix may be advertised along with an overlay next-hop that can be a GW IP address or an ESI, or without an overlay next-hop, in which case the BGP next-hop will point at the egress NVE and the MAC in the Router's MAC Extended Community will provide the inner MAC destination address to be used. As discussed throughout the document, the existing EVPN RT-2 does not meet the requirements for all the DC use cases, therefore a new EVPN route type is required.

This new EVPN route type 5 decouples the IP Prefix advertisements from the MAC route advertisements in EVPN, hence:

a) Allows the clean and clear advertisements of ipv4 or ipv6 prefixes in an NLRI with no MAC addresses in the route key, so that only IP information is used in BGP route comparisons.

b) Since the route type is different from the MAC/IP advertisement route, the advertisement of prefixes will be excluded from all the procedures defined for the advertisement of VM MACs, e.g. MAC Mobility or aliasing. As a result of that, the current EVPN procedures do not need to be modified.

c) Allows a flexible implementation where the prefix can be linked to different types of next-hops: overlay IP address, overlay ESI, underlay IP next-hops, etc.

d) An EVPN implementation not requiring IP Prefixes can simply discard them by looking at the route type value. An unknown route type MUST be ignored by the receiving NVE/PE.

7. 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].

8. Security Considerations

9. IANA Considerations

10. References

10.1 Normative References

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

10.2 Informative References

[EVPN] Sajassi et al., "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-11.txt, work in progress, October, 2014

[EVPN-OVERLAYS] Sajassi-Drake et al., "A Network Virtualization Overlay Solution using EVPN", draft-sd-l2vpn-evpn-overlay-03.txt, work in progress, June, 2014

[EVPN-INTERSUBNET] Sajassi et al., "IP Inter-Subnet Forwarding in EVPN", draft-sajassi-l2vpn-evpn-inter-subnet-forwarding-05.txt, work in progress, October, 2014

11. Acknowledgments

The authors would like to thank Mukul Katiyar and Senthil Sathappan for their valuable feedback and contributions. The following people also helped improving this document with their feedback: Antoni Przygienda and Thomas Morin.

12. Authors' Addresses

Jorge Rabadan
Alcatel-Lucent
777 E. Middlefield Road
Mountain View, CA 94043 USA
Email: jorge.rabadan@alcatel-lucent.com

Wim Henderickx
Alcatel-Lucent
Email: wim.henderickx@alcatel-lucent.com

Florin Balus

Nuage Networks
Email: florin@nuagenetworks.net

Aldrin Isaac
Bloomberg
Email: aisaac71@bloomberg.net

Senad Palislamovic
Alcatel-Lucent
Email: senad.palislamovic@alcatel-lucent.com

John E. Drake
Juniper Networks
Email: jdrake@juniper.net

Ali Sajassi
Cisco
Email: sajassi@cisco.com

INTERNET-DRAFT
Intended Status: Standards track
Expires: January 5, 2015

R. Fernando
D. Rao
Cisco
L. Fang
Microsoft
M. Napierala
AT&T
N. So
Vinci Systems
A. Farrel
Juniper Networks

July 4, 2014

Virtual Topologies for Service Chaining in BGP/IP MPLS VPNs

draft-rfernando-l3vpn-service-chaining-04

Abstract

This document presents techniques built upon BGP/IP MPLS VPN control plane mechanisms to construct virtual topologies for service chaining. These virtual service topologies interconnect network zones and constrain the flow of traffic between these zones via a sequence of service nodes so that service functions can be applied to the traffic.

This document also describes approaches enabled by both the routing control plane and by network orchestration to realize these virtual service topologies.

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."

The list of current Internet-Drafts can be accessed at

http://www.ietf.org/1id-abstracts.html

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

Table of Contents

1. Introduction

Network topologies and routing design in enterprise, data center, and campus networks typically reflect the needs of the organization in terms of performance, scale, security, and availability. For scale and security reasons, these networks may be composed of multiple small domains or zones each serving one or more functions of the organization.

A network zone is a logical grouping of physical assets that supports certain applications. Hosts can communicate freely within a zone. That is, a datagram traveling between two hosts in the same zone is not routed through any servers that examine the datagram payload and apply services (such as security or load balancing) to the traffic. But a datagram traveling between hosts in different zones may be subject to additional services to meet the needs of scaling, performance, and security for the applications or the networks themselves.

Networks have achieved division into zones and the imposition of services through a combination of physical topology constraints and routing. For example, one can force datagrams to go through a firewall (FW) by putting the FW in the physical data path from a source to the destination, or by causing the routed path form source to destination to go via a FW that would not normally be on the path. Similarly, the datagrams may need to go through a security gateway for security services, or a Load Balancer (LB) for load balancing services.

In virtualized data centers, appliances, applications, and network functions, including IP VPN provider edge (PE) and customer edge (CE) functions are all commonly virtualized. That is, they exist as software instances residing in servers or appliances instead of individual (dedicated) physical devices.

Migrating a network with all its functions and infrastructure elements to realization in a virtualized data center requires network overlay mechanisms that provide the ability to create virtual network topologies that mimic physical networks, and that provide the ability to constrain the flow of routing and traffic over these virtual network topologies.

A data center uses a virtual topology in which the servers are in the "virtual" data path, rather than in the physical data path. For example, a traffic flow might previously have had the source PE-1 and destination at an Autonomous System Border Router (ASBR), ASBR-1, and the flow might have needed to be serviced by FW-1 and LB-1. In this virtualized data center, the functions of all four nodes could be

provided by virtual nodes that could be placed at arbitrary locations across the data center. Thus the "virtual service chain" vPE-1, FW-1, vLB-1, vASBR-1, that is the sequence of virtual service nodes that packet must traverse, could be realized by a logical path between arbitrary physical locations in the data center.

A data center will likely support multiple tenants. A tenant is a customer who uses the virtualized data center services. Each tenant might require different connectedness (i.e., a different virtual topology) between their zones and applications, and might need the ability to apply different network policies such that the services for inter-zone traffic are applied in a specific order according to the organization objectives of the tenant. Furthermore, a data center might need multiple virtual topologies per tenant to handle different types of application traffic.

Additionally, a data center operator may choose to provide services for multiple tenants on the same virtualized end device, for example, a server. Such multi-tenant devices must utilize techniques such as routing isolation to retain separation between tenants' traffic.

To address all of these requirements, the mechanisms devised for use in a data center need to be flexible enough to accommodate the custom needs of the tenants and their applications, and at the same time must be robust enough to satisfy the scale, performance, and high availability needs that are demanded by the operator of the virtual network infrastructure that has a very large number of tenants each with different application types, large networks, multiple services, and high-volume traffic.

Toward this end, this document introduces the concept of virtual service topologies and extends IP MPLS VPN control plane mechanisms to constrain routing and traffic flow over virtual service topologies.

The creation of these topologies and the setting up of the forwarding tables to steer traffic over them may be carried out either by extensions to IP MPLS VPN procedures and functionality at the PEs, or via a "software defined networking" (SDN) approach. This document specifies the use of both approaches, but uses the IP MPLS VPN option to illustrate the various steps involved.

1.1 Terminology

This document uses the following acronyms and terms.

Terms	Meaning
AS	Autonomous System
ASBR	Autonomous System Border Router
CE	Customer Edge
FW	Firewall
I2RS	Interface to the Routing System
L3VPN	Layer 3 VPN
LB	Load Balancer
NLRI	Network Layer Reachability Information [RFC4271]
P	Provider backbone router
proxy-arp	proxy-Address Resolution Protocol
RR	Route Reflector
RT	Route Target
SDN	Software Defined Network
vCE	virtual Customer Edge router [I-D.fang-l3vpn-virtual-ce]
vFW	virtual Firewall
vLB	virtual Load Balancer
VM	Virtual Machine
vPC	virtual Private Cloud
vPE	virtual Provider Edge router [I-D.fang-l3vpn-virtual-pe]
VPN	Virtual Private Network
VRF	VPN Routing and Forwarding table [RFC4364]
vRR	virtual Route Reflector

This document also uses the following general terms:

Service-PE:
A BGP/IP MPLS VPN PE to which a service node in a virtual service topology is attached. The PE directs incoming traffic from other PEs or from attached hosts to the service node via an MPLS VPN label or IP lookup. The PE also forwards traffic from the service node to the next node in the chain. A Service-PE is a logical entity and a given PE may be attached to both a service node and an application host VM.

Service node:
A physical or virtual service appliance/application which inspects and/or redirects the flow of inter-zone traffic. Examples of service nodes include FWs, LBs, and deep packet inspectors. The service node acts as a CE in the VPN network.

Service chain: A sequence of service nodes that interconnect the zones containing the source and destination hosts or endpoints. The service chain is unidirectional and creates a one way traffic flow between source zone and destination zone.

Virtual service topology:
A virtual service topology consists of a sequence of service-PEs and their attached service nodes created in a specific order. A service topology is constructed via one or more routes that direct the traffic flow among the PEs that form the service chain.

Service-topology-RT:
A BGP route attribute that identifies the specific service topology.

Tenant:
A tenant is a higher-level management construct. In the control/ forwarding plane it is the collection of various virtual networks that get instantiated. A tenant may have more than one virtual network or VPN.

Zone:
A logical grouping of physical or virtual assets that supports certain applications or a subset thereof. VMs or hosts can communicate freely within a zone.

2. Intra-Zone Routing and Traffic Forwarding

This section provides a brief overview of how the BGP/IP MPLS VPN [RFC4364] control plane can be used in a DC network to used to divide the network into a number of zones. The subsequent sections in the document build on this base model to create inter-zone service topologies by interconnecting these zones and forcing inter-zone traffic to travel through a sequence of servers where the sequence of servers depends on the tuple <source zone, destination zone, application>.

The notion of a BGP/IP VPN when applied to the virtual data center works in the following manner.

The VM that runs the applications in the server is treated as a CE attached to the VPN. A CE/VM belongs to a zone. The PE is the first hop router from the CE/VM and the PE-CE link is single hop from a layer-3 perspective. Any of the available physical, logical or tunneling technologies can be used to create this "direct" link between the CE/VM and its attached PE(s).

If a PE attaches to one or more CEs of a certain zone, the PE must

have exactly one VRF for that zone, and the PE-CE links to those CEs must all be associated with that VRF. Intra-zone connectivity between CE/VMs that attach to different PEs is achieved by designating an RT per zone (zone-RT) that is both an import RT and an export RT of all PE VRFs that terminate the CE/VMs that belong to the zone. A VM may have multiple virtual interfaces that attach to different zones.

It is further assumed that the CE/VMs are associated with network policies that are activated on an attached PE when a CE/VM is instantiated. These policies dictate how the network is set up for the CE/VM including the properties of the CE-PE link, the IP address of the CE/VM, the zones to which it belongs, QoS policies, etc. There are many ways to accomplish this step, but a description of such mechanisms is outside the scope of this document.

When the CE/VM is activated, the attached PE starts to export the CEs IP address with the corresponding zone-RT. This allows unrestricted any-to-any communication between the newly active VM and the rest of the VMs in the zone.

The classification of VMs into a zone is driven by the communication and security policy and is independent of the addressing scheme for the VMs. The VMs in a zone may be in the same or different IP subnets with user-defined mask-lengths. The PE advertises /32 routes to advertise reachability to locally attached VMs. If two VMs are in the same IP subnet, the PE may employ proxy-ARP to assist the VM to resolve ARP for other VMs in the IP subnet, and may use IP forwarding to carry traffic between the VMs. When a VM is attached to a remote PE, IP VPN forwarding is used to tunnel packets to the remote PE.

3. Inter-Zone Routing and Traffic Forwarding

A simple form of inter-zone traffic forwarding can be achieved using extranets or hub-and-spoke L3VPN configurations [RFC7024]. However, the ability to enforce constrained traffic flows through a set of services is non-existent in extranets and is limited in hub-and-spoke setups.

Note that the inter-zone services cannot always be assumed to reside and be in-lined on a PE. There is a need to virtualize the services themselves so that they can be implemented on commodity hardware and scaled out 'elastically' when traffic demands increase. This creates a situation where services for traffic between zones may be applied not only at the source-zone PE or the destination-zone PE. Mechanisms are required that make it easy to direct inter-zone traffic through the appropriate set of service nodes that might be remote or virtualized.

3.1 Traffic Forwarding Operational Flow

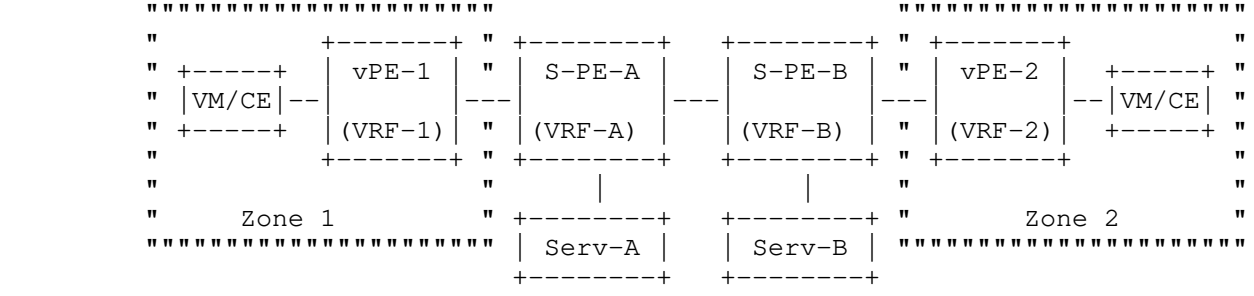
Traffic from a source endpoint (a VM/CE) in a source zone reaches an ingress zone-PE and is associated with a VRF in that zone as described above. The zone-PE will forward the traffic and direct it toward the first service-node. If the service-node is attached to the zone-PE, the zone-PE will forward the packets out of one of its access interfaces. If the service-node is attached to a different service-PE, the zone-PE will encapsulate the packets and send them toward the service-PE. The zone-PE and service PE may be connected via an intermediate network of devices and the encapsulation causes the packets to be tunneled across this intermediate network.

The service-PE will receive these encapsulated packets from the source zone-PE, decapsulate them, and forward them to its attached service-node. The traffic that comes back to the service-PE from the service-node must now be forwarded to the next service-node in the chain. As above, the next service-node may be locally attached or at a remote service-PE.

At the last service-PE in the chain, the traffic that comes back from a service-node must be forwarded to the destination in the target zone. Just as with the service-nodes, the destination may be attached to the service-PE or reachable via another PE.

As can be seen from this description, a given packet flow needs to be forwarded differently at each PE depending on whether it is arriving from a node attached to the PE or from a remote PE, and depending on whether the traffic is to be routed toward a node attached to the PE or attached to a remote PE. The next-hop for a flow changes depending on the relative position within the service chain.

Figure 1 illustrates a virtual service topology, where hosts in Zone 1 are interconnected with hosts in Zone 2 via two service nodes (Serv-A and Serv-B) attached to two service-PEs (S-PE-A and S-PE-B respectively).



The different forwarding paths can be achieved at any PE as follows.

- o Each service node is associated with two VRFs at the service PE to which it is attached: an in-VRF for traffic toward the service node, and an out-VRF for traffic from the service node.

- o Traffic for the in-VRF arrives from the previous node in the service chain, and traffic for the out-VRF is destined toward the next node in the service chain, or toward the destination zone.

- o The in-VRF has one or more routes with a next-hop of a local access interface where the service node is attached. The out-VRF has routes with a next-hop of the next service node, which may be situated locally on the service-PE or at a remote PE.

The installation of the forwarding entries to implement the flow described above may be achieved either via IP VPN mechanisms described in Sections 4 and 5, or using an SDN approach, as described in Section 6.

4. Inter-Zone Model

The inter-zone model has the following steps.

4.1 Constructing the Virtual Service Topology

The virtual service topology described in the previous section is constructed via one or more service routes that direct the traffic flow among the PEs forming the service chain. There should be a route per service node. The service topologies, and hence the service routes, are constructed on a per-VPN basis. This service topology is independent of the routes for the actual destination for a flow, i.e., the addresses of the VMs present in the various zones. There can be multiple service topologies for a given VPN.

4.1.1 Reachability to the Service Nodes

Each service node is identified by an IP address that is scoped within the VPN. The service node is also associated with an in-VRF and out-VRF at the attached service node.

Reachability to the various service nodes in the service chain occurs via regular BGP/IP VPN route advertisements.

A service-PE will export a route for each service node attached to it. Each route will contain the Route-Target configured for the VPN, and a forwarding label that is associated with the logical in-VRF for to directly forward incoming traffic from the other PEs to the

service node.

The routes to reach the various service nodes are imported into and installed in each out-VRF at a service-PE, as well as in the zone VRF on the ingress zone-PE.

4.1.2 Provisioning the Service Chain

At each PE supporting a given VPN, the sequence of service nodes in a service chain can be specified in a VPN service route policy.

To create the service chain and give it a unique identity, each PE may be provisioned with the following tuple for every service chain that it belongs to:

{Service-topology-RT, Service-node-Sequence}

where Service-node-Sequence is simply an ordered list of the service node IP addresses that are in the chain.

Every service chain has a single unique service-topology-RT that is provisioned in all participating PEs.

A PE will also be provisioned with the tables and/or other configuration that support the various zones and the in- and out-VRFs for the services.

4.1.3 Zone Prefix Next-Hop Resolution

Routes representing hosts, VMs or other destinations associated with a zone are called zone prefixes. A zone prefix will have its regular zone RTs attached when it is originated. This will be used by PEs that have VRFs for the same zone to import these prefixes to enable direct communication between VMs in the same zone.

In addition to the intra-zone RTs, zone prefixes are also tagged at the point of origination with the set of Service-topology-RTs to which they belong.

Since they are tagged with the Service-topology-RT, zone prefixes get imported into the VRFs of the service-PEs that form the service chain associated to that topology RT. Note that the Service-topology-RT was added to the relevant VRF's import RT list during the virtual topology construction phase. These routes may be installed in the in-VRF and out-VRF at the service-PEs as well as in the ingress zone's VRF.

Note that the approach being described introduces a change in the

behavior of the service-PEs and ingress zone's PEs compared to normal BGP VPN behavior, but does not require protocol changes to BGP. This modification to PE behavior allows the automatic and constrained flow of traffic via the service chain.

The PE, based on the presence of the Service-topology-RT in the zone routes it receives, will perform the following actions:

1. It will ignore the next-hop and VPN label that were advertised in the NLRI.

2. Instead, it will select the appropriate Service next-hop from the Service-node sequence associated with the Service-topology-RT. In the out-VRF associated with a service node, it will select the next service node in the sequence.

3. It will further resolve this Service next-hop IP address locally in the associated VRF, instead of in the global routing table. It will use the next-hop (and label, if remote) associated with this IP address to encapsulate traffic toward the next service node.

4. If the importing service-PE is the last service-PE, it uses the next hop that came with the zone prefix for route resolution. It also uses the VPN label that came with the prefix.

In this way the zone prefixes in the intermediate service-PE hops recurse over the service chain forcing the traffic destined to them to flow through the virtual service topology.

Traffic for the zone prefix goes through the service hops created by the service topology. At each service hop, the service-PE directs the traffic to the service node. Once the service node is done processing the traffic, it sends it back to the service-PE which forwards the traffic to the next service-PE, and so on.

A significant benefit of this next-hop indirection is to avoid redundant advertisement of zone prefixes from the end-zone or service-PEs. Also, when the virtual service topology is changed (due to addition or removal of service nodes), there should be no change to the zone prefix's import/export RT configuration, and hence no re-advertisement of zone prefixes.

There should be one service topology RT per virtual service topology. There can be multiple virtual service topologies and hence service topology RTs for a given VPN.

Virtual service topologies are constructed unidirectionally. Traffic in opposite directions between the same pair of zones will be

supported by two different service topologies and hence two service topology routes. These two service topologies might or might not be symmetrical, i.e. they might or might not traverse the same sequence As noted above, a service node route is advertised with a label that directs incoming traffic to the attached service node. Alternatively, an aggregate label may be used for the service route and an IP route lookup done in the in-VRF at the service-PE to send traffic to the service node.

Note that a new service node could be inserted into the service chain seamlessly by just configuring the service policy appropriately.

4.2 Per-VM Service Chains

While the service-topology-RT allows an efficient inheritance of the service chain for all VMs or prefixes in a zone, there may be a need to create a distinct service chain for an individual VM or prefix. This may be done by provisioning a separate service-topology RT and service node sequence. The VM route carries the service-topology RT, and the destination and service PEs are provisioned with this RT as described above.

5. Routing Considerations

5.1 Multiple Service Topologies

A service-PE can support multiple distinct service topologies for a VPN.

5.2 Multipath

One could use all tools available in BGP to constrain the propagation and resolution of state created by the service topology [RFC4684].

Additional service nodes can be introduced to scale out a particular service. Each such service would be represented by a virtual IP address, and multiple service nodes associated with it. Multiple service-PEs may advertise a route to this address based on the presence of an attached service node instance, thereby creating multiple equal cost paths. This technique could be used to elastically scale out the service nodes with traffic demand.

5.3 Supporting Redundancy

For stateful services an active-standby mechanism could be used at the service level. In this case, the inter-zone traffic should prefer the active service node over the standby service node.

At a routing level, this is achieved by setting up two paths for the same service route: one path goes through the active service node and the other through the standby service node. The active service path can then be made to win over the standby service path by appropriately setting the BGP path attributes of the service topology route such that the active path succeeds in path selection. This forces all inter-zone traffic through the active service node.

5.4 Route Aggregation

Instead of the actual zone prefixes being imported and used at various points along the chain, the zone prefixes may be aggregated at a specific PE and the aggregate zone prefix used in the service chain between zones. In such a case, it is the aggregate zone prefix that carries the service-topology-RT and gets imported in the service-PEs that comprise the service chain.

6. Orchestration Driven Approach

In an orchestration driven approach, there is no need for the zone or service PEs to determine the appropriate next-hops based on the specified service node sequence. All the necessary policy computations are carried out, and the forwarding tables for the various VRFs at the PEs determined, by a central orchestrator or controller.

The orchestrator communicates with the various PEs (typically virtual PEs on the end-servers) to populate the forwarding tables.

The protocol used to communicate between the controller/orchestration and the PE/vPE must be a standard, programmatic interface. There are several possible options to this programmatic interface, some being under discussion in the IETF's Interface to Routing Systems (I2RS) initiative, [I-D.ietf-i2rs-architecture], [I-D.ietf-i2rs-problem-statement]. One specific option is defined in [IPSE].

7. Security Considerations

To be added.

8. Management Considerations

To be added.

9. IANA Considerations

This proposal does not have any IANA implications.

10. Acknowledgements

The authors would like to thank the following individuals for their review and feedback on the proposal: Eric Rosen, Jim Guichard, Paul Quinn, Peter, Bosch, David Ward, Ashok Ganesan. The option of configuring an ordered sequence of service nodes via policy is derived from a suggestion from Eric Rosen.

11. References

11.1 Normative References

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

11.2 Informative References

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[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, November 2006.

[RFC7024] Jeng, H., Uttaro, J., Jalil, L., Decraene, B., Rekhter, Y., and R. Aggarwal, "Virtual Hub-and-Spoke in BGP/MPLS VPNs", RFC 7024, October 2013.

[I-D.fang-l3vpn-virtual-ce] L. Fang, et al.,"BGP/MPLS IP VPN Virtual CE", draft-fang-l3vpn-virtual-ce, work in progress.

[I-D.fang-l3vpn-virtual-pe] L. Fang, et al., "BGP/MPLS IP VPN Virtual PE", draft-fang-l3vpn-virtual-pe, work in progress.

[I-D.ietf-i2rs-architecture] Atlas, A., Halpern, J., Hares, S., Ward, D., and T Nadeau, "An Architecture for the Interface to the Routing System", draft-ietf-i2rs-architecture, work in progress.

[I-D.ietf-i2rs-problem-statement] Atlas, A., Nadeau, T., and D. Ward, "Interface to the Routing System Problem Statement", draft-ietf-i2rs-problem-statement, work in progress.

[IPSE]
Fernando, R., Boutros, S., Rao, D., "Interface to a Packet Switching Element", draft-rfernando-ipse-00, work in progress.

Authors' Addresses

Dhananjaya Rao
Cisco
170 W Tasman Dr
San Jose, CA
Email: dhrao@cisco.com

Rex Fernando
Cisco
170 W Tasman Dr
San Jose, CA
Email: rex@cisco.com

Luyuan Fang
Microsoft
5600 148th Ave NE
Redmond, WA 98052
Email: lufang@microsoft.com

Maria Napierala
AT&T
200 Laurel Avenue
Middletown, NJ 07748
Email: mnapierala@att.com

Ning So
Vinci Systems, Inc.
Email: ningso@yahoo.com

Adrian Farrel
Juniper Networks
Email: adrian@olddog.co.uk

L2VPN Workgroup
INTERNET-DRAFT
Intended Status: Standards Track

Ali Sajassi
Samer Salam
Sami Boutros
Cisco

Wim Henderickx
Jorge Rabadan
Alcatel-Lucent

Jim Uttaro
AT&T

Aldrin Isaac
Bloomberg

Expires: April 27, 2015

October 27, 2014

E-TREE Support in EVPN & PBB-EVPN
draft-sajassi-bess-evpn-etree-00

Abstract

The Metro Ethernet Forum (MEF) has defined a rooted-multipoint Ethernet service known as Ethernet Tree (E-Tree). [ETREE-FMWK] proposes a solution framework for supporting this service in MPLS networks. This document discusses how those functional requirements can be easily met with EVPN.

Table of Contents

1 Introduction

The Metro Ethernet Forum (MEF) has defined a rooted-multipoint Ethernet service known as Ethernet Tree (E-Tree). In an E-Tree service, endpoints are labeled as either Root or Leaf sites. Root sites can communicate with all other sites. Leaf sites can communicate with Root sites but not with other Leaf sites.

[ETREE-FMWK] proposes the solution framework for supporting E-Tree service in MPLS networks. The document identifies the functional components of the overall solution to emulate E-Tree services in addition to Ethernet LAN (E-LAN) services on an existing MPLS network.

[EVPN] is a solution for multipoint L2VPN services, with advanced multi-homing capabilities, using BGP for distributing customer/client MAC address reach-ability information over the MPLS/IP network. [PBB-EVPN] combines the functionality of EVPN with [802.1ah] Provider Backbone Bridging for MAC address scalability.

This document discusses how the functional requirements for E-Tree service can be easily met with EVPN and PBB-EVPN.

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].

2 E-Tree Scenarios and EVPN / PBB-EVPN Support

In this section, we will categorize support for E-Tree into three different scenarios, depending on the nature of the site association (Root/Leaf) per PE or per Ethernet Segment:

- Leaf OR Root site(s) per PE

- Leaf AND Root site(s) per PE

- Leaf AND Root site(s) per Ethernet Segment

2.1 Scenario 1: Leaf OR Root site(s) per PE

In this scenario, a PE may have Root sites OR Leaf sites for a given VPN instance, but not both concurrently. The PE may have both Root and Leaf sites albeit for different VPNs. Every Ethernet Segment

connected to the PE is uniquely identified as either a Root or a Leaf site.

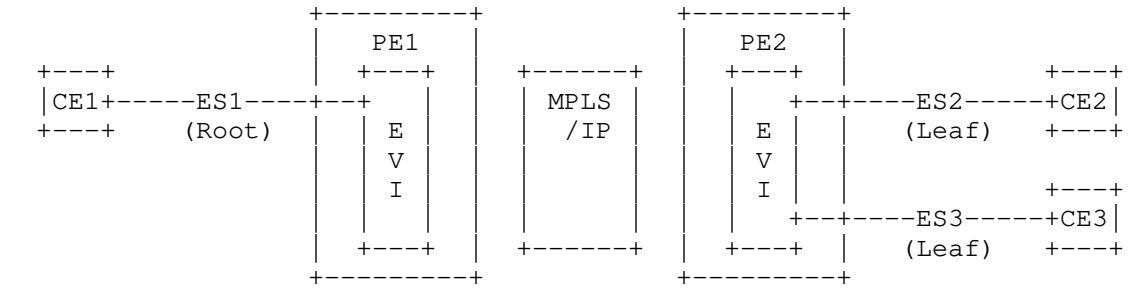


Figure 1: Scenario 1

2.2 Scenario 2: Leaf AND Root site(s) per PE

In this scenario, a PE may have a set of one or more Root sites AND a set of one or more Leaf sites for a given VPN instance. Every Ethernet Segment connected to the PE is uniquely identified as either a Root or a Leaf site.

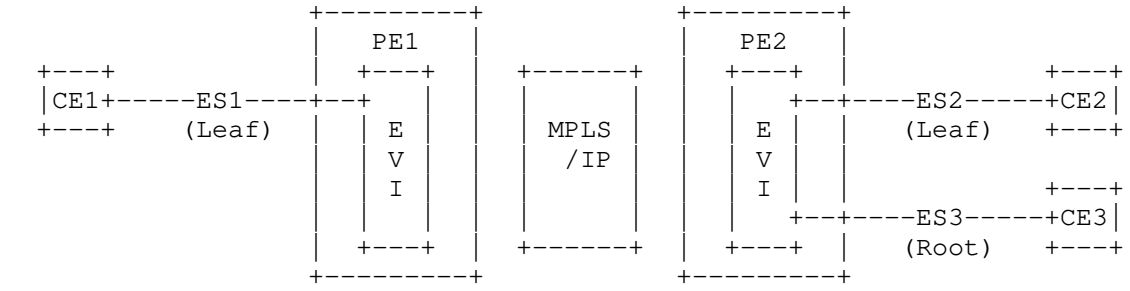


Figure 2: Scenario 2

2.3 Scenario 3: Leaf AND Root site(s) per Ethernet Segment

In this scenario, a PE may have a set of one or more Root sites AND a set of one or more Leaf sites for a given VPN instance. An Ethernet Segment connected to the PE may be identified as both a Root and a Leaf site concurrently.

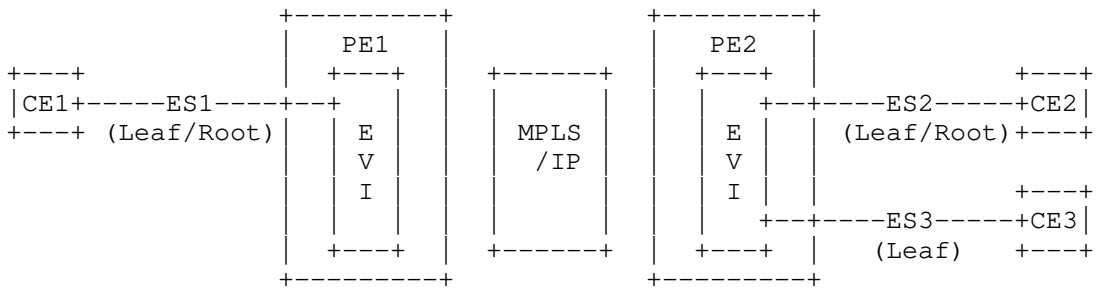


Figure 3: Scenario 3

3 Operation for EVPN

[EVPN] defines the notion of an Ethernet Segment which can be readily used to identify a Root and/or Leaf site in E-TREE services. In other words, [EVPN] has inherent capability to support E-TREE services without defining any new BGP routes. It only requires a minor modification to the existing procedures and a modification to a BGP attribute as shown in this section.

The following procedures are used consistently for all the scenarios highlighted in the previous section.

3.1 Known Unicast Traffic

For known unicast traffic, the PE must advertise a Root/Leaf indication along with each MAC Advertisement route, to indicate whether the associated MAC address was learnt from a Root or a Leaf. This enables remote PEs to perform ingress filtering for known unicast traffic: On the ingress PE, the MAC destination address lookup yields, in addition to the forwarding adjacency, a flag which indicates whether the target MAC is associated with a Root or a Leaf site. The ingress PE cross-checks this flag with the status of the originating site, and if both are a Leaf, then the packet is not forwarded.

The PE places all Leaf Ethernet Segments of a given bridge domain in a single split-horizon group in order to prevent intra-PE forwarding among Leaf segments. This split-horizon function applies to BUM traffic as well.

To support the above ingress filtering functionality, a new Root/Leaf indication flag will added to the Tunnel Encapsulation Type Extended Community [RFC5512]. This extended community will be advertised with each EVPN MAC Advertisement route.

3.2 BUM Traffic

For BUM traffic, it is not possible to perform filtering on the ingress PE, as is the case with known unicast, because of the multi-destination nature of the traffic. As such, the solution relies on egress filtering. In order to apply the proper egress filtering, which varies based on whether a packet is sent from a Root or a Leaf, the MPLS-encapsulated frames MUST be tagged with an indication of whether they originated from a Root or a Leaf Ethernet Segment. This can be achieved in EVPN through the use of the ESI MPLS label, since this label identifies the Ethernet Segment of origin of a given frame. The egress PE determines whether or not to forward a particular frame to an Ethernet Segment depending on the split-horizon rule defined in [EVPN]:

- If the ESI Label indicates that the source Ethernet Segment is a Root, then the frame can be forwarded on a segment granted that it passes the split-horizon check.

- If the ESI Label indicates that the source Ethernet Segment is a Leaf, then the frame can be forwarded only on a Root segment, granted that it passes the split-horizon check.

When advertising the ESI MPLS label for a given Ethernet Segment, a PE must indicate whether the corresponding ESI is a Root or a Leaf site. This can be done by encoding the Root or Leaf indication in the Flags field of the ESI MPLS label Extended Community attribute ([EVPN] Section 8) to indicate Root/Leaf status.

In the case where a multi-homed Ethernet Segment has both Root and Leaf sites attached, two ESI MPLS labels are allocated and advertised: one ESI MPLS label denotes Root and the other denotes Leaf. The ingress PE imposes the right ESI MPLS label depending on whether the Ethernet frame originated from the Root or Leaf site on that Ethernet Segment. The mechanism by which the PE identifies whether a given frame originated from a Root or Leaf site on the segment is based on the Ethernet Tag associated with the frame. Other mechanisms of identification, beyond the Ethernet Tag, are outside the scope of this document. It should be noted that support for both Root and Leaf sites on a single Ethernet Segment requires that the PE performs the Ethernet Segment split-horizon check on a per Ethernet Tag basis. In the case where a multi-homed Ethernet Segment has either Root or Leaf sites attached, then a single ESI MPL label is allocated and advertised.

Furthermore, a PE advertises two special ESI MPLS labels: one for Root and another for Leaf. These are used by remote PEs for traffic originating from single-homed segments and for multi-homed segments

that are not connected to the advertising PE. Note that these special labels are advertised on a per PE basis (i.e. each PE advertises only two such special labels).

In addition to egress filtering (which is a MUST requirement), an EVPN PE implementation MAY provide topology constraint among the PEs belonging to the same EVI associated with an E-TREE service. The purpose of this topology constraint is to avoid having PEs with only host Leaf sites importing and processing BGP MAC routes from each other, thereby unnecessarily exhausting their RIB tables. However, as soon as a Root site is added to a Leaf PE, then that PE needs to process MAC routes from all other Leaf PEs and add them to its forwarding table. To support such topology constrain in EVPN, two BGP Route-Targets (RTs) are used for every EVPN Instance (EVI): one RT is associated with the Root sites and the other is associated with the Leaf sites. On a per EVI basis, every PE exports the single RT associated with its type of site(s). Furthermore, a PE with Root site(s) imports both Root and Leaf RTs, whereas a PE with Leaf site(s) only imports the Root RT. If for a given EVI, the PEs can eventually have both Leaf and Root sites attached, even though they may start as Root-only or Leaf-only PEs, then it is recommended to use a single RT per EVI and avoid additional configuration and operational overhead. If the number of EVIs is very large (e.g., more than 32K or 64K), then RT type 0 as defined in [RFC4360] SHOULD be used; otherwise, RT type 2 is sufficient.

3.3 E-TREE Traffic Flows for EVPN

Per [ETREE-FMWK], a generic E-Tree service supports all of the following traffic flows:

- Ethernet Unicast from Root to Roots & Leaf
- Ethernet Unicast from Leaf to Root
- Ethernet Broadcast/Multicast from Root to Roots & Leafs
- Ethernet Broadcast/Multicast from Leaf to Roots

A particular E-Tree service may need to support all of the above types of flows or only a select subset, depending on the target application. In the case where unicast flows need not be supported, the L2VPN PEs can avoid performing any MAC learning function.

In the subsections that follow, we will describe the operation of EVPN to support E-Tree service with and without MAC learning.

3.3.1 E-Tree with MAC Learning

The PEs implementing an E-Tree service must perform MAC learning when

unicast traffic flows must be supported from Root to Leaf or from Leaf to Root sites. In this case, the PE with Root sites performs MAC learning in the data-path over the Ethernet Segments, and advertises reachability in EVPN MAC Advertisement routes. These routes will be imported by PEs that have Leaf sites as well as by PEs that have Root sites, in a given EVI. Similarly, the PEs with Leaf sites perform MAC learning in the data-path over their Ethernet Segments, and advertise reachability in EVPN MAC Advertisement routes which are imported only by PEs with at least one Root site in the EVI. A PE with only Leaf sites will not import these routes. PEs with Root and/or Leaf sites may use the Ethernet A-D routes for aliasing (in the case of multi-homed segments) and for mass MAC withdrawal.

To support multicast/broadcast from Root to Leaf sites, either a P2MP tree rooted at the PE(s) with the Root site(s) or ingress replication can be used. The multicast tunnels are set up through the exchange of the EVPN Inclusive Multicast route, as defined in [EVPN].

To support multicast/broadcast from Leaf to Root sites, ingress replication should be sufficient for most scenarios where there is a single Root or few Roots. If the number of Roots is large, a P2MP tree rooted at the PEs with Leaf sites may be used.

3.3.2 E-Tree without MAC Learning

The PEs implementing an E-Tree service need not perform MAC learning when the traffic flows between Root and Leaf sites are multicast or broadcast. In this case, the PEs do not exchange EVPN MAC Advertisement routes. Instead, the Ethernet A-D routes are used to exchange the EVPN labels.

The fields of the Ethernet A-D route are populated per the procedures defined in [EVPN], and the route import rules are as described in previous sections.

4 Operation for PBB-EVPN

In PBB-EVPN, the PE must advertise a Root/Leaf indication along with each MAC Advertisement route, to indicate whether the associated B-MAC address corresponds to a Root or a Leaf site. Similar to the EVPN case, this flag will be added to the Tunnel Encapsulation Type Extended Community [RFC5512], and advertised with each MAC Advertisement route.

In the case where a multi-homed Ethernet Segment has both Root and Leaf sites attached, two B-MAC addresses are allocated and advertised: one B-MAC address denotes Root and the other denotes Leaf. The ingress PE uses the right B-MAC source address depending on

whether the Ethernet frame originated from the Root or Leaf site on that Ethernet Segment. The mechanism by which the PE identifies whether a given frame originated from a Root or Leaf site on the segment is based on the Ethernet Tag associated with the frame. Other mechanisms of identification, beyond the Ethernet Tag, are outside the scope of this document. It should be noted that support for both Root and Leaf sites on a single Ethernet Segment requires that the PE performs the Ethernet Segment split-horizon check on a per Ethernet Tag basis. In the case where a multi-homed Ethernet Segment has either Root or Leaf sites attached, then a single B-MAC address is allocated and advertised per segment.

Furthermore, a PE advertises two global B-MAC addresses: one for Root and another for Leaf, and tags them as such in the MAC Advertisement routes. These B-MAC addresses are used as source addresses for traffic originating from single-homed segments.

4.1 Known Unicast Traffic

For known unicast traffic, the PEs perform ingress filtering: On the ingress PE, the C-MAC destination address lookup yields, in addition to the target B-MAC address and forwarding adjacency, a flag which indicates whether the target B-MAC is associated with a Root or a Leaf site. The ingress PE cross-checks this flag with the status of the originating site, and if both are a Leaf, then the packet is not forwarded.

The PE places all Leaf Ethernet Segments of a given bridge domain in a single split-horizon group in order to prevent intra-PE forwarding among Leaf segments. This split-horizon function applies to BUM traffic as well.

4.2 BUM Traffic

For BUM traffic, the PEs must perform egress filtering. When a PE receives a MAC advertisement route, it updates its Ethernet Segment egress filtering function (based on the B-MAC source address), as follows:

- If the MAC Advertisement route indicates that the advertised B-MAC is a Leaf, and the local Ethernet Segment is a Leaf as well, then the source B-MAC address is added to the B-MAC filtering list.

- Otherwise, the B-MAC filtering list is not updated.

When the egress PE receives the packet, it examines the B-MAC source address to check whether it should filter or forward the frame. Note that this uses the same filtering logic as baseline [PBB-EVPN] and

does not require any additional flags in the data-plane.

5 Acknowledgement

We would like to thank Sami Boutros and Dennis Cai for their comments.

6 Security Considerations

Same security considerations as [EVPN].

7 IANA Considerations

Allocation of Extended Community Type and Sub-Type for EVPN.

8 References

8.1 Normative References

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

[RFC4360] S. Sangli et al, ""BGP Extended Communities Attribute", February, 2006.

[RFC5512] Mohapatra, P. and E. Rosen, "The BGP Encapsulation Subsequent Address Family Identifier (SAFI) and the BGP Tunnel Encapsulation Attribute", RFC 5512, April 2009.

8.2 Informative References

[ETREE-FMWK] Key et al., "A Framework for E-Tree Service over MPLS Network", draft-ietf-l2vpn-etree-frwk-03, work in progress, September 2013.

[EVPN] Sajassi et al., "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-04.txt, work in progress, July, 2013.

[PBB-EVPN] Sajassi et al., "PBB-EVPN", draft-ietf-l2vpn-pbb-evpn-05.txt, work in progress, October, 2013.

Authors' Addresses

Ali Sajassi
Cisco
Email: sajassi@cisco.com

Samer Salam
Cisco
Email: ssalam@cisco.com

Wim Henderickx
Alcatel-Lucent
Email: wim.henderickx@alcatel-lucent.com

Jim Uttaro
AT&T
Email: ju1738@att.com

Aldrin
Bloomberg Issac
Email: aisaac71@bloomberg.net

Sami Boutros
Cisco
Email: sboutros@cisco.com

Internet Working Group
Internet Draft
Category: Standards Track

A. Sajassi
P. Brissette
Cisco
R. Schell
Verizon
J. Drake
Juniper
J. Rabadan
Nokia

Expires: August 26, 2016 February 26, 2018

EVPN Virtual Ethernet Segment
draft-sajassi-bess-evpn-virtual-eth-segment-03

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."

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

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

Abstract

EVPN and PBB-EVPN introduce a family of solutions for multipoint Ethernet services over MPLS/IP network with many advanced capabilities among which their multi-homing capabilities. These solutions define two types of multi-homing for an Ethernet Segment (ES): 1) Single-Active and 2) All-Active, where an Ethernet Segment is defined as a set of links between the multi-homed device/network and the set of PE devices that they are connected to.

Some Service Providers want to extend the concept of the physical links in an ES to Ethernet Virtual Circuits (EVCs) where many of such EVCs can be aggregated on a single physical External Network-to-Network Interface (ENNI). An ES that consists of a set of EVCs instead of physical links is referred to as a virtual ES (vES). This draft describes the requirements and the extensions needed to support vES in EVPN and PBB-EVPN.

Conventions

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.

Table of Contents

1. Introduction

[EVPN] and [PBB-EVPN] introduce a family of solutions for multipoint Ethernet services over MPLS/IP network with many advanced capabilities among which their multi-homing capabilities. These solutions define two types of multi-homing for an Ethernet Segment (ES): 1) Single-Active and 2) All-Active, where an Ethernet Segment is defined as a set of links between the multi-homed device/network and the set of PE devices that they are connected to.

This document extends the Ethernet Segment concept so that an ES can be associated to a set of EVCs or other objects such as MPLS Label Switch Paths (LSP) or Pseudowires (PW).

1.1 Virtual Ethernet Segments in Access Ethernet Networks

Some Service Providers (SPs) want to extend the concept of the physical links in an ES to Ethernet Virtual Circuits (EVCs) where many of such EVCs can be aggregated on a single physical External Network-to-Network Interface (ENNI). An ES that consists of a set of EVCs instead of physical links is referred to as a virtual ES (vES). Figure below depicts two PE devices (PE1 and PE2) each with an ENNI where a number of vES's are aggregated on - each of which through its associated EVC.

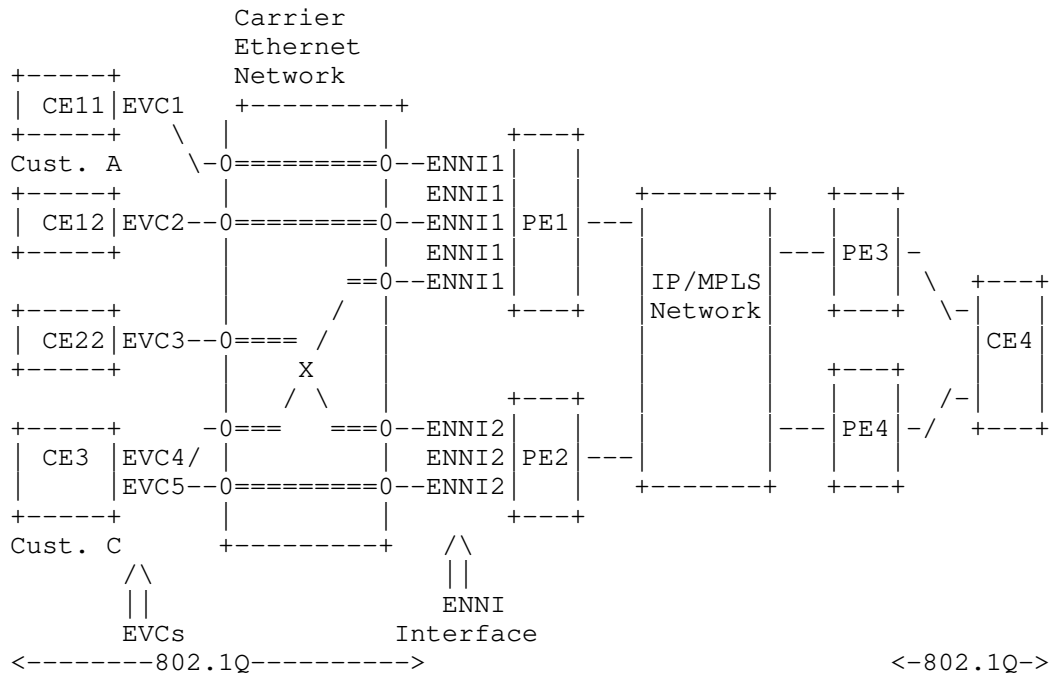


Figure 1: DHD/DHN (both SA/AA) and SH on same ENNI

E-NNIs are commonly used to reach off-network / out-of-franchise customer sites via independent Ethernet access networks or third-party Ethernet Access Providers (EAP) (see above figure). E-NNIs can aggregate traffic from hundreds to thousands of vES's; where, each vES is represented by its associated EVC on that ENNI. As a result, ENNIs and their associated EVCs are a key element of SP off-networks that are carefully designed and closely monitored.

In order to meet customer's Service Level Agreements (SLA), SPs build redundancy via multiple E-PEs / ENNIs (as shown in figure above) where a given vES can be multi-homed to two or more PE devices (on two or more ENNIs) via their associated EVCs. Just like physical ES's in [EVPN] and [PBB-EVPN] solutions, these vES's can be single-homed or multi-homed ES's and when multi-homed, then can operate in either Single-Active or All-Active redundancy modes. In a typical SP off-network scenario, an ENNI can be associated with several thousands of single-homed vES's, several hundreds of Single-Active vES's and it may also be associated with tens or hundreds of All-Active vES's.

1.2 Virtual Ethernet Segments in Access MPLS Networks

Other Service Providers (SPs) want to extend the concept of the physical links in an ES to individual Pseudowires (PW) or to MPLS Label Switched Paths (LSPs) per [EVPN-VPWS] in Access MPLS networks. Figure 2 illustrates this concept.

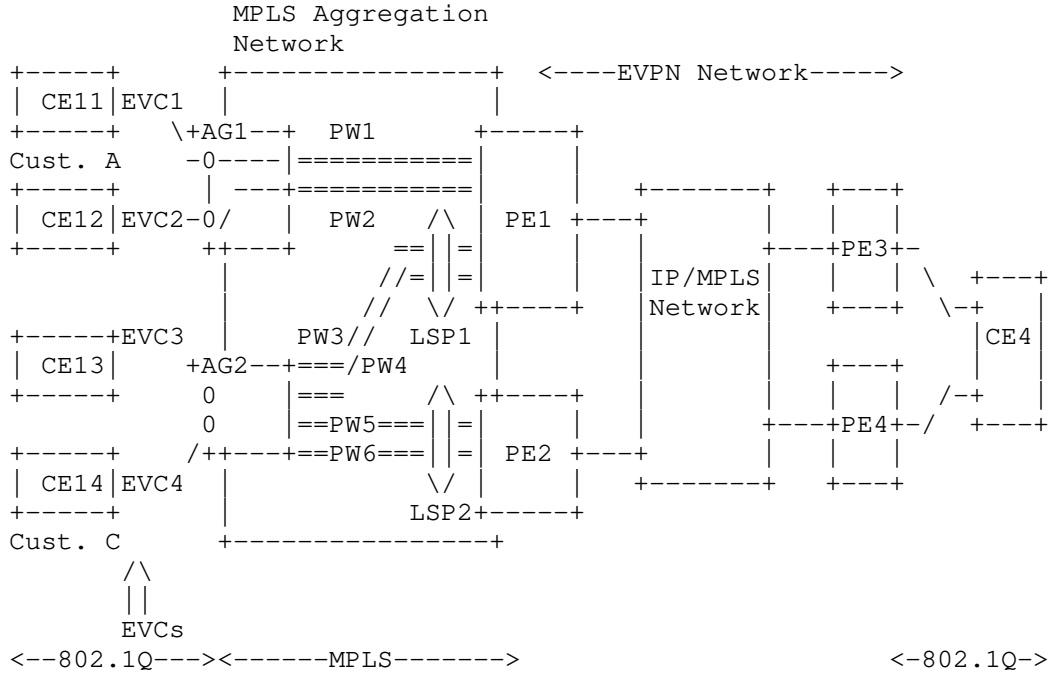


Figure 2: DHN and SH on Access MPLS networks

In some cases, Service Providers use Access MPLS Networks that belong to separate administrative entities or third parties as a way to get access to the their own IP/MPLS network infrastructure. This is the case illustrated in Figure 2.

An ES is defined as a set of individual PWs if they cannot be aggregated into a common LSP. If the aggregation of PWs is possible, the ES can be associated to an LSP in a given PE. In the example of Figure 2, EVC3 is connected to a VPWS instance in AG2 that is connected to PE1 and PE2 via PW3 and PW5 respectively. EVC4 is connected to a separate VPWS instance on AG2 that gets connected to

an EVI on PE1 and PE2 via PW4 and PW6, respectively. Since the PWs for the two VPWS instances can be aggregated into the same LSPs going to the EVPN network, a common virtual ES can be defined for LSP1 and LSP2. This ES will be shared by two separate EVIs in the EVPN network.

In some cases, this aggregation of PWs into common LSPs may not be possible. For instance, if PW3 were terminated into a third PE, e.g. PE3, instead of PE1, the ES would need to be defined on a per individual PW on each PE, i.e. PW3 and PW5 would belong to ES-1, whereas PW4 and PW6 would be associated to ES-2.

An ES that consists of a set of LSPs or individual PWs is also referred as virtual ES (vES) in this document."

This draft describes requirements and the extensions needed to support vES in [EVPN] and [PBB-EVPN]. Section 3 lists the set of requirements for Virtual ES's. Section 4 describes the solution for [PBB-EVPN] to meet these requirements. Section 5 describes the failure handling and recovery for Virtual ES's in [PBB-EVPN]. Section 6 covers scalability and fast convergence required for Virtual ES's in [PBB-EVPN].

2. Terminology

AC: Attachment Circuit
BEB: Backbone Edge Bridge
B-MAC: Backbone MAC Address
CE: Customer Edge
CFM: Connectivity Fault Management
C-MAC: Customer/Client MAC Address
DHD: Dual-homed Device
DHN: Dual-homed Network
ENNI: External Network-Network Interface
ES: Ethernet Segment
ESI: Ethernet-Segment Identifier
EVC: Ethernet Virtual Circuit
EVPN: Ethernet VPN
LACP: Link Aggregation Control Protocol
PE: Provider Edge
SH: Single-Homed

Single-Active Redundancy Mode (SA): 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 (AA): 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.

3. Requirements

This section describes the requirements specific to virtual Ethernet Segment (vES) for (PBB-)EVPN solutions. These requirements are in addition to the ones described in [EVPN-REQ], [EVPN], and [PBB-EVPN].

3.1. Single-Homed & Multi-Homed Virtual Ethernet Segments

A PE needs to support the following types of vES's:

(R1a) A PE MUST handle single-homed vES's on a single physical port (e.g., single ENNI)

(R1b) A PE MUST handle a mix of Single-Homed vES's and Single-Active multi-homed vES's simultaneously on a single physical port (e.g., single ENNI). Single-Active multi-homed vES's will be simply referred to as Single-Active vES's through the rest of this document.

(R1c) A PE MAY handle All-Active multi-homed vES's on a single physical port. All-Active multi-homed vES's will be simply referred to as All-Active vES's through the rest of this document.

(R1d) A PE MAY handle a mixed of All-Active vES's along with other types of vES's on a single physical port

(R1e) A Multi-Homed vES (Single-Active or All-Active) can be spread across any two or more PEs (on two or more ENNIs)

3.2. Scalability

A single physical port (e.g., ENNI) can be associated with many vES's. The following requirements give a quantitative measure for each vES type.

(R2a) A PE MUST handle thousands or tens of thousands of Single-homed vES's on a single physical port (e.g., single ENNI)

(R2b) A PE MUST handle hundreds of Single-Active vES's on a single physical port (e.g., single ENNI)

(R2c) A PE MAY handle tens or hundreds of All-Active Multi-Homed

vES's on a single physical port (e.g., single ENNI)

(R2d) A PE MUST handle the above scale for a mix of Single-homed vES's and Single-Active vES's simultaneously on a single physical port (e.g., single ENNI)

(R4e) A PE MAY handle the above sale for a mixed of All-Active Multi-Homed vES's along with other types of vES's on a single physical port

3.3. Local Switching

Many vES's of different types can be aggregated on a single physical port on a PE device and some of these vES can belong to the same service instance (or customer). This translates into the need for supporting local switching among the vES's of the same service instance on the same physical port (e.g., ENNI) of the PE.

(R3a) A PE MUST support local switching among different vES's belonging to the same service instance (or customer) on a single physical port. For example, in the above figure (1), PE1 MUST support local switching between CE11 and CE12 (both belonging to customer A) that are mapped to two Single-homed vES's on ENNI1.

In case of Single-Active vES's, the local switching is performed among active EVCs belonging to the same service instance on the same ENNI.

3.4. EVC Service Types

A physical port (e.g., ENNI) of a PE can aggregate many EVCs each of which is associated with a vES. Furthermore, an EVC may carry one or more VLANs. Typically, an EVC carries a single VLAN and thus it is associated with a single broadcast domain. However, there is no restriction on an EVC to carry more than one VLANs.

(R4a) An EVC can be associated with a single broadcast domain – e.g., VLAN-based service or VLAN bundle service

(R4b) An EVC MAY be associated with several broadcast domains – e.g., VLAN-aware bundle service

In the same way, a PE can aggregated many LSPs and PWs. In the case of individual PWs per vES, typically a PW is associated with a single broadcast domain, but there is no restriction on the PW to carry more than one VLAN if the PW is defined as vc-type VLAN.

(R4c) A PW can be associated with a single broadcast domain – e.g., VLAN-based service or VLAN bundle service.

(R4b) An PW MAY be associated with several broadcast domains – e.g., VLAN-aware bundle service."

3.5. Designated Forwarder (DF) Election

Section 8.5 of [EVPN] describes the default procedure for DF election in EVPN which is also used in [PBB-EVPN]. This default DF election procedure is performed at the granularity of <ESI, EVI>. In case of a vES, the same EVPN default procedure for DF election also applies; however, at the granularity of <vESI, EVI>; where vESI is the virtual Ethernet Segment Identifier. As in [EVPN], this default procedure for DF election at the granularity of <vESI, EVI> is also referred to as "service carving"; where, EVI is represented by an I-SID in PBB-EVPN and by a EVI service-id/vpn-id in EVPN. With service carving, it is possible to evenly distribute the DFs for different vES's among different PEs, thus distributing the traffic among different PEs. The following list the requirements apply to DF election of vES's for EVPN.

(R5a) A vES with m EVCs can be distributed among n ENNIs belonging to p PEs in any arbitrary oder; where n >= P >= m. For example, if there is an vES with 2 EVCs and there are 5 ENNIs on 5 PEs (PE1 through PE5), then vES can be dual-homed to PE2 and PE4 and the DF election must be performed between PE2 and PE4.

(R5b) Each vES MUST be identified by its own virtual ESI (vESI)

3.6. OAM

In order to detect the failure of individual EVC and perform DF election for its associated vES as the result of this failure, each EVC should be monitored independently.

(R6a) Each EVC SHOULD be monitored for its health independently

(R6b) A single EVC failure (among many aggregated on a single physical port/ENNI) MUST trigger DF election for its associated vES.

3.7. Failure & Recovery

(R7a) Failure and failure recovery of an EVC for a Single-homed vES SHALL NOT impact any other EVCs for its own service instance or any other service instances. In other words, for PBB-EVPN, it SHALL NOT trigger any MAC flushing both within its own I-SID as well as other I-SIDs.

(R7b) In case of All-Active Multi-Homed vES, failure and failure

recovery of an EVC for that vES SHALL NOT impact any other EVCs for its own service instance or any other service instances. In other words, for PBB-EVPN, it SHALL NOT trigger any MAC flushing both within its own I-SID as well as other I-SIDs.

(R7c) Failure & failure recovery of an EVC for a Single-Active vES SHALL only impact its own service instance. In other words, for PBB-EVPN, MAC flushing SHALL be limited to the associated I-SID only and SHALL NOT impact any other I-SIDs.

(R7d) Failure & failure recovery of an EVC for a Single-Active vES MAY only impact C-MACs associated with MHD/MHNs for that service instance. In other words, MAC flushing SHOULD be limited to single service instance (I-SID in the case of PBB-EVPN) and only CMACs for Single-Active MHD/MHNs.

3.8. Fast Convergence

Since large number of EVCs (and their associated vES's) are aggregated via a single physical port (e.g., ENNI), then the failure of that physical port impacts large number of vES's and triggers large number of ES route withdrawals. Formulating, sending, receiving, and processing such large number of BGP messages can introduce delay in DF election and convergence time. As such, it is highly desirable to have a mass-withdraw mechanism similar to the one in the [EVPN] for withdrawing large number of Ethernet A-D routes.

(R8a) There SHOULD be a mechanism equivalent to EVPN mass-withdraw such that upon an ENNI failure, only a single BGP message is needed to indicate to the remote PEs to trigger DF election for all impacted vES associated with that ENNI.

4. Solution Overview

The solutions described in [EVPN] and [PBB-EVPN] are leveraged as is with one simple modification and that is the ESI assignment is performed for a group of EVCs instead of a group of links. In other words, the ESI is associated with a virtual ES (vES) and that's why it will be referred to as vESI.

For EVPN solution, everything basically remains the same except for the handling of physical port failure where many vES's can be impacted. Section 5.1 and 5.3 below describe the handling of physical port/link failure for EVPN. In a typical multi-homed operation, MAC addresses are learned behind a vES are advertised with the ESI corresponding to the vES (i.e., vESI). EVPN aliasing and mass-withdraw operations are performed with respect to vES. In other

words, the Ethernet A-D routes for these operations are advertised with vESI instead of ESI.

For PBB-EVPN solution, the main change is with respect to the BMAC address assignment which is performed similar to what is described in section 7.2.1.1 of [PBB-EVPN] with the following refinements:

- One shared BMAC address is used per PE for the single-homed vES's. In other words, a single BMAC is shared for all single-homed vES's on that PE.

- One shared BMAC address should be used per PE per physical port (e.g., ENNI) for the Single-Active vES's. In other words, a single BMAC is shared for all Single-Active vES's that shared the same ENNI.

- One shared BMAC address can be used for all Single-Active vES's on that PE.

- One BMAC address is used per EVC per physical port per PE for each All-Active multi-homed vES. In other words, a single BMAC address is used per vES for All-Active multi-homing scenarios.

- A single BMAC address may also be used per vES per PE for Single-Active multi-homing scenarios.

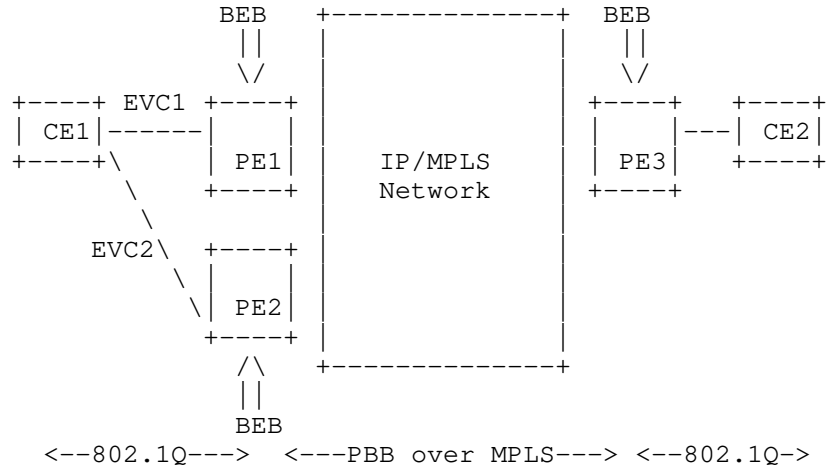


Figure 2: PBB-EVPN Network

4.1. EVPN DF Election for vES

The procedure for service carving for virtual Ethernet Segments is the same as the one outlined in section 8.5 of [EVPN] except for the fact that ES is replaced with vES. For the sake of clarity and completeness, this procedure is repeated below:

1. When a PE discovers the ESI or is configured with the ESI associated with its attached vES, it advertises an Ethernet Segment route with the associated ES-Import extended community attribute.

2. The PE then starts a timer (default value = 3 seconds) to allow the reception of Ethernet Segment routes from other PE nodes connected to the same vES. This timer value MUST be same across all PEs connected to the same vES.

3. When the timer expires, each PE builds an ordered list of the IP addresses of all the PE nodes connected to the vES (including itself), in increasing numeric value. Each IP address in this list is extracted from the "Originator Router's IP address" field of the advertised Ethernet Segment route. Every PE is then given an ordinal indicating its position in the ordered list, starting with 0 as the ordinal for the PE with the numerically lowest IP address. The ordinals are used to determine which PE node will be the DF for a given EVPN instance on the vES using the following rule: Assuming a redundancy group of N PE nodes, the PE with ordinal i is the DF for an EVPN instance with an associated EVI ID value of V when (V mod N) = i.

It should be noted that using "Originator Router's IP address" field in the Ethernet Segment route to get the PE IP address needed for the ordered list, allows for a CE to be multi-homed across different ASes if such need ever arises.

4. The PE that is elected as a DF for a given EVPN instance will unblock traffic for that EVPN instance. Note that the DF PE unblocks all traffic in both ingress and egress directions for Single-Active vES and unblocks multi-destination in egress direction for All-Active Multi-homed vES. All non-DF PEs block all traffic in both ingress and egress directions for Single-Active vES and block multi-destination traffic in the egress direction for All-Active multi-homed vES.

In the case of an EVC failure, the affected PE withdraws its Ethernet Segment route. This will re-trigger the service carving procedures on all the PEs in the RG. For PE node failure, or upon PE commissioning or decommissioning, the PEs re-trigger the service carving across all affected vES's. In case of a Single-Active multi-homing, when a service moves from one PE in the RG to another PE as a result of re-carving, the PE, which ends up being the elected DF for the service, SHOULD trigger a MAC address flush notification towards the

associated vES. This can be done, for e.g. using IEEE 802.1ak MVRP 'new' declaration.

For LSP and PW based vES, the non-DF PE SHOULD signal PW-status 'standby' signaling to the AG PE, and the new DF MAY send an LDP MAC withdraw message as a MAC address flush notification.

5. Failure Handling & Recovery

There are a number of failure scenarios to consider such as:

A: CE Uplink Port Failure
B: Ethernet Access Network Failure
C: PE Access-facing Port or link Failure
D: PE Node Failure
E: PE isolation from IP/MPLS network

[EVPN] and [PBB-EVPN] solutions provide protection against such failures as described in the corresponding references. In the presence of virtual Ethernet Segments (vES's) in these solutions, besides the above failure scenarios, there is one more scenario to consider and that is EVC failure. This implies that individual EVCs need to be monitored and upon their failure detection, appropriate DF election procedures and failure recovery mechanism need to be executed.

[ETH-OAM] is used for monitoring EVCs and upon failure detection of a given EVC, DF election procedure per section [4.1] is executed. For PBB-EVPN, some addition extensions are needed to failure handling and recovery procedures of [PBB-EVPN] in order to meet the above requirements. These extensions are describe in the next section.

[MPLS-OAM] and [PW-OAM] are used for monitoring the status of LSPs and/or PWs associated to vES.

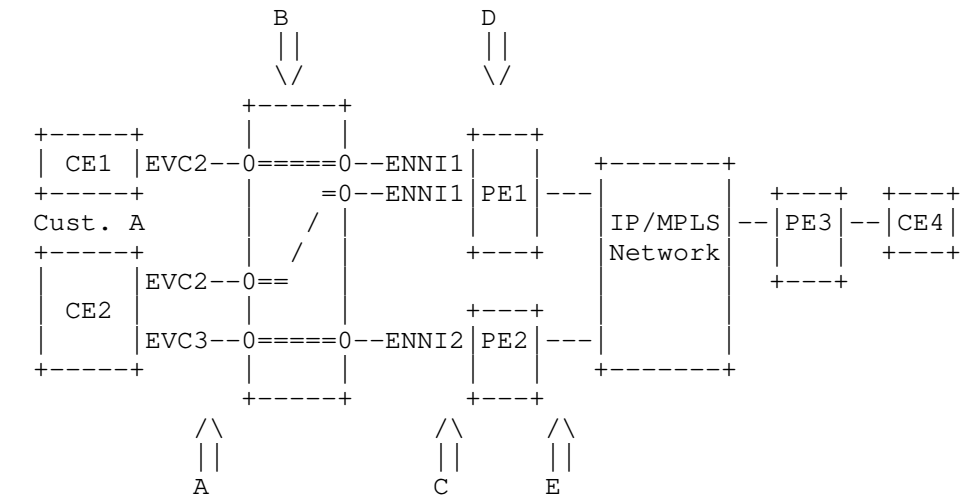


Figure 3: Failure Scenarios A,B,C,D and E

5.1. Failure Handling for Single-Active vES in EVPN

When a PE connected to a Single-Active multi-homed Ethernet Segment loses connectivity to the segment, due to link or port failure, it signals the remote PE to flush all CMAC addresses associated with that Ethernet Segment. This is done by advertising a mass-withdraw message using Ethernet A-D per-ES route. To be precise, there is no MAC flush per-se if there is only one backup PE for a given ES - i.e., only an update of the forwarding entries per backup-path procedure in [RFC 7432].

In case of an EVC failure that impacts a single vES, the exact same EVPN procedure is used. In this case, the message using Ethernet A-D per ES route carries the vESI representing the vES which is in turn associated with the failed EVC. The remote PEs upon receiving this message perform the same procedures outlined in section 8.2 of [EVPN].

5.2. EVC Failure Handling for Single-Active vES in PBB-EVPN

When a PE connected to a Single-Active multi-homed Ethernet Segment loses connectivity to the segment, due to link or port failure, it signals the remote PE to flush all CMAC addresses associated with that Ethernet Segment. This is done by advertising a BMAC route along with MAC Mobility Extended community.

In case of an EVC failure that impacts a single vES, if the above

PBB-EVPN procedure is used, it results in excessive CMAC flushing because a single physical port can support large number of EVCs (and their associated vES's) and thus advertising a BMAC corresponding to the physical port with MAC mobility Extended community will result in flushing CMAC addresses not just for the impacted EVC but for all other EVCs on that port.

In order to reduce the scope of CMAC flushing to only the impacted service instances (the service instance(s) impacted by the EVC failure), the BGP flush message is sent along with a list of impacted I-SID(s) represented by the new EVPN I-SID Extended Community as defined in section 6. Since typically an EVC maps to a single broadcast domain and thus a single service instance, the list only contains a single I-SID. However, if the failed EVC carries multiple VLANs each with its own broadcast domain, then the list contains several I-SIDs - one for each broadcast domain. This new BGP flush message basically instructs the remote PE to perform flushing for CMACs corresponding to the advertised BMAC only across the advertised list of I-ISIDs (which is typically one).

The above BMAC route that is advertised with the MAC Mobility Extended Community, can either represent the MAC address of the physical port that the failed EVC is associated with, or it can represent the MAC address of the PE. In the latter case, this is the dedicated MAC address used for all Single-Active vES's on that PE. The former one performs better than the latter one in terms of reducing the scope of flushing as described below and thus it is the recommended approach.

Advertising the BMAC route that represent the physical port (e.g., ENNI) on which the failed EVC reside along with MAC Mobility and I-SID extended communities provide the most optimum mechanism for CMAC flushing upon EVC failure in PBB-EVPN for Single-Active vES because:

1) Only CMAC addresses for the impacted service instances are flushed.

2) Only a subset of CMAC addresses for the impacted service instances are flushed - only the ones that are learned over the BMAC associated with the failed EVC. In other words, only a small fraction of the CMACs for the impacted service instance(s) are flushed.

5.3. Port Failure Handling for Single-Active vES's in EVPN

When a large number of EVCs are aggregated via a single physical port on a PE; where each EVC corresponds to a vES, then the port failure impacts all the associated EVCs and their corresponding vES's. If the

number of EVCs corresponding to the Single-Active vES's for that physical port is in thousands, then thousands of service instances are impacted. Therefore, the BGP flush message need to be inclusive of all these impacted service instances. In order to achieve this, the following extensions are added to the baseline EVPN mechanism:

1) A PE when advertises an Ether-AD per ES route for a given vES, it colors it with the MAC address of the physical port which is associated with that vES. The receiving PEs take note of this color and create a list of vES's for this color.

2) Upon a port failure (e.g., ENNI failure), the PE advertise a special mass-withdraw message with the MAC address of the failed port (i.e., the color of the port) encoded in the ESI field. For this encoding, type 3 ESI is used with the MAC field set to the MAC address of the port and the 3-octet local discriminator field set to 0xFFFFFF. This mass-withdraw route is advertised with a list of Route Targets corresponding to the impacted service instances. If the number of Route Targets is more than they can fit into a single attribute, then a set of Ethernet A-D per ES routes are advertised. The remote PEs upon receiving this message, realize that this is a special mass-withdraw message and they access the list of the vES's for the specified color. Next, they initiate mass-withdraw procedure for each of the vES's in the list.

5.4. Port Failure Handling for Single-Active vES's in PBB-EVPN

When a large number of EVCs are aggregated via a single physical port on a PE; where each EVC corresponds to a vES, then the port failure impacts all the associated EVCs and their corresponding vES's. If the number of EVCs corresponding to the Single-Active vES's for that physical port is in thousands, then thousands of service instances (I-SIDs) are impacted. Therefore, the BGP flush message need to be sent with a list of thousands of I-SIDs. The new I-SID Extended Community provides a way to encode upto 24 I-SIDs in each Extended Community if the impacted I-SIDs are sequential (the base I-SID value plus the next 23 I-SID values). So, the packing efficiency can range from 1 to 24 and there can be up to 400 such Extended Community sent along with a BGP flush message for a total of 400 to 9600 I-SIDs. If the number of I-SIDs is large enough to not fit in a single Attribute, then either a number of BGP flush messages (with different RDs) can be transmitted or a single BGP flush message without the I-SID list can be transmitted. If the BGP flush message is transmitted without the I-SID list, then it instructs the receiving PEs to flush CMACs associated with that BMAC across all I-SIDs. For simplicity, we opt for the latter option in this document. In other words, if the number of impacted I-SIDs exceed that of a single BGP flush message,

then the flush message is sent without the I-SID list.

As also described in [PBB-EVPN], there are two ways to signal flush message upon a physical port failure:

1) If the MAC address of the physical port is used for PBB encapsulation as BMAC SA, then upon the port failure, the PE MUST use the EVPN MAC route withdrawal message to signal the flush

2) If the PE shared MAC address is used for PBB encapsulation as BMAC SA, then upon the port failure, the PE MUST re-advertise this MAC route with the MAC Mobility Extended Community to signal the flush

The first method is recommended because it reduces the scope of flushing the most.

5.5. Fast Convergence in PBB-EVPN

As described above, when a large number of EVCs are aggregated via a physical port on a PE; where each EVC corresponds to a vES, then the port failure impacts all the associated EVCs and their corresponding vES's. Two actions must be taken as the result of such port failure:

- Flushing of all CMACs associated with the BMAC of the failed port for the impacted I-SIDs

- DF election for all impacted vES's associated with the failed port

Section 5.4 describes how to flush CMAC address in the most optimum way – e.g., to flush least number of CMAC addresses for the impacted I-SIDs. This section describes how to perform DF election in the most optimum way – e.g., to trigger DF election for all impacted vES's (which can be in thousands) among the participating PEs via a single BGP message as opposed to sending thousands of BGP messages – one per vES.

In order to devise such fast convergence mechanism that can be triggered via a single BGP message, all vES's associated with a given physical port (e.g., ENNI) are colored with the same color representing that physical port. The MAC address of the physical port is used for this coloring purposes and when the PE advertises an ES route for a vES associated with that physical port, it advertises it with an EVPN MAC Extended Community indicating the color of that port.

The receiving PEs take note of this color and for each such color,

they create a list of vES's associated with this color (with this MAC address). Now, when a port failure occurs, the impacted PE needs to notify the other PEs of this color so that these PEs can identify all the impacted vES's associated with that color (from the above list) and re-execute DF election procedures for all the impacted vES's.

In PBB-EVPN, there are two ways to convey this color to other PEs upon a port failure - one corresponding to each method for signaling flush message as described in section 5.4. If for PBB encapsulation, the MAC address of the physical port is used as BMAC SA, then upon the port failure, the PE sends MAC withdrawal message with the MAC address of the failed port as the color. However, if for PBB encapsulation, the shared MAC address of the PE (dedicated for all Single-Active vES's) is used as BMAC SA, then upon the port failure, the PE re-advertises the MAC route (that carries the shared BMAC) along with this new EVPN MAC Extended Community to indicate the color along with MAC Mobility Extended Community.

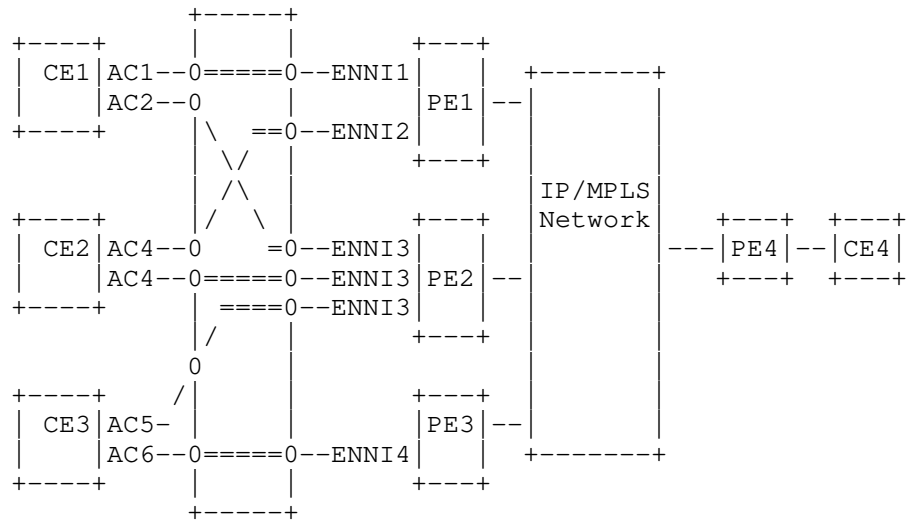


Figure 4: Fast Convergence Upon ENNI Failure

The following describes the procedure for coloring vES's and fast convergence using this color in more details:

1- When a vES is configured, the PE colors the vES with the MAC address of the corresponding physical port and advertises the Ethernet Segment route for this vES with this color.

2- All other PEs (in the redundancy group) take note of this color and add the vES to the list for this color.

3- Upon the occurrence of a port failure (e.g., an ENNI failure), the PE sends the flush message in one of the two ways described above indicating this color.

4- On reception of the flush message, other PEs use this info to flush their impacted CMACs and to initiate DF election procedures across all their affected vES's.

5- The PE with the physical port failure (ENNI failure), also send ES route withdrawal for every impacted vES's. The other PEs upon receiving these messages, clear up their BGP tables. It should be noted the ES route withdrawal messages are not used for executing DF election procedures by the receiving PEs.

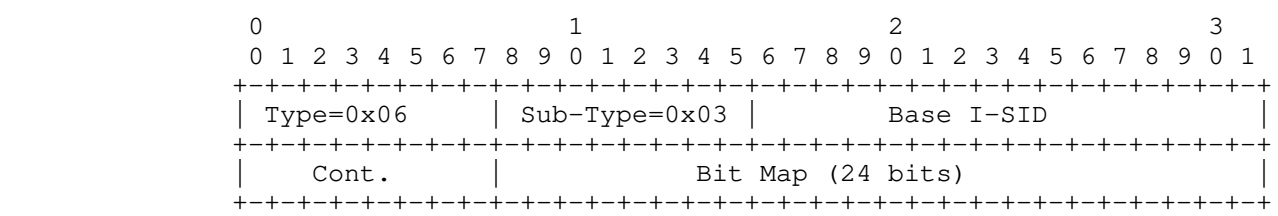
6. BGP Encoding

This document defines one new BGP Extended Community for EVPN.

6.1. I-SID Extended Community

A new EVPN BGP Extended Community called I-SID is introduced. This new extended community is a transitive extended community with the Type field of 0x06 (EVPN) and the Sub-Type of 0x04.

The I-SID Extended Community is encoded as an 8-octet value as follows:



This extended community is used to indicate the list of I-SIDs associated with a given Ethernet Segment.

24-bit map represents the next 24 I-SID after the base I-SID. For example based I-SID of 10025 with 24-bit map of zero means, only a single I-SID of 10025. I-SID of 10025 with bit map of 0x000001 means there are two I-SIDs, 10025 and 10026.

7. Acknowledgements

TBD

8. Security Considerations This document does not introduce any additional security constraints.

9. IANA Considerations

TBD

10. Intellectual Property Considerations

This document is being submitted for use in IETF standards discussions.

11. Normative References

[PBB] Clauses 25 and 26 of "IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", IEEE Std 802.1Q, 2013.

12. Informative References

[RFC7209] Sajassi, et al., "Requirements for Ethernet VPN (EVPN)", RFC7209, May 2014.

[EVPN] Sajassi, et al., "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-07.txt, work in progress, May 7, 2014.

[PBB-EVPN] Sajassi, et al., "PBB-EVPN", draft-ietf-l2vpn-pbb-evpn-07.txt, work in progress, June 18, 2014.

13. Authors' Addresses

Ali Sajassi
Cisco Systems
Email: sajassi@cisco.com

Patrice Brissette
Cisco Systems
Email: pbrisset@cisco.com

Rick Schell
Verizon
Email: richard.schell@verizon.com

John E Drake
Juniper
Email: jdrake@juniper.net

Tapraj Singh
Juniper
Email: tsingh@juniper.net

Jorge Rabadan
ALU
Email: jorge.rabadan@alcatel-lucent.com

INTERNET-DRAFT
Intended Status: Standard Track

Ali Sajassi
Samer Salam
Cisco

Nick Del Regno
Verizon

Jorge Rabadan
Alcatel-Lucent

Expires: April 27, 2015

October 27, 2014

(PBB-)EVPN Seamless Integration with (PBB-)VPLS
draft-sajassi-bess-evpn-vpls-seamless-integ-00

Abstract

This draft discusses the backward compatibility of the (PBB-)EVPN solution with (PBB-)VPLS and provides mechanisms for seamless integration of the two technologies in the same MPLS/IP network on a per-VPN-instance basis.

Table of Contents

1 Introduction

VPLS and PBB-VPLS are widely-deployed L2VPN technologies. Many SPs who are looking at adopting EVPN and PBB-EVPN want to preserve their investment in the (PBB-)VPLS networks. Hence, it is required to provide mechanisms by which (PBB-)EVPN technology can be introduced into existing L2VPN networks without requiring a fork-lift upgrade. This document discusses mechanisms for the seamless integration of the two technologies in the same MPLS/IP network.

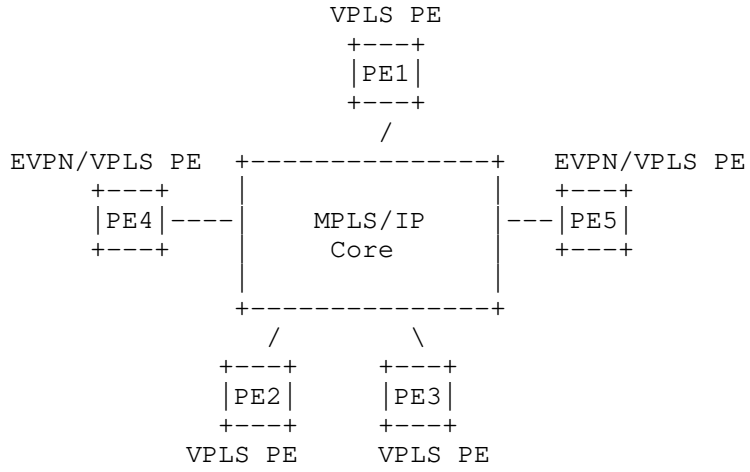


Figure 1: Seamless Integration of (PBB-)EVPN PEs & (PBB-)VPLS

Section 2 provides the details of the requirements. Section 3 discusses PBB-VPLS integration with PBB-EVPN. Section 4 discusses the integration of VPLS and EVPN. Section 5 discusses the integration of VPLS and PBB-EVPN, and finally Section 6 discusses the solution advantages.

It is worth noting that the scenario where PBB-VPLS is integrated with EVPN, is for future study and upon market validation. The reason for that is that deployments which employ PBB-VPLS typically require PBB encapsulation for various reasons. Hence, it is expected that for those deployments the evolution path would be from PBB-VPLS towards PBB-EVPN, rather than EVPN.

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].

2. Requirements

Following are the key requirements for backward compatibility between (PBB-)EVPN and (PBB-)VPLS:

1. The solution MUST allow for staged migration towards (PBB-)EVPN on a site-by-site basis per VPN instance - e.g., new EVPN sites to be provisioned on (PBB-)EVPN PEs.

2. The solution MUST require no changes to existing VPLS or PBB-VPLS PEs, not even a software upgrade.

3. The solution MUST allow for the coexistence of PE nodes running (PBB-)EVPN and (PBB-)VPLS for the same VPN instance and single-homed segments.

4. The solution MUST support single-active redundancy of multi-homed networks and multi-homed devices for (PBB-)EVPN PEs.

5. In case of single-active redundancy, the participant VPN instances MAY span across both (PBB-)EVPN PEs and (PBB-)VPLS PEs as long as single-active redundancy is employed by (PBB-)EVPN PEs. In case of an ES link failure, the (PBB-)EVPN PEs will send a BGP mass-withdraw to the EVPN peers OR MAC advertisement with MAC Mobility extended community for PBB-EVPN AND an LDP MAC withdrawal to the VPLS peers.

6. The solution SHOULD support all-active redundancy of multi-homed networks and multi-homed devices for (PBB-)EVPN PEs.

7. In case of all-active redundancy, the participant VPN instances SHOULD be confined to (PBB-)EVPN PEs only.

These requirements collectively allow for the seamless insertion of the (PBB-)EVPN technology into brown-field (PBB-)VPLS deployments.

3 PBB-VPLS Integration with PBB-EVPN

In order to support seamless integration with (PBB-)VPLS, the (PBB-)EVPN PEs MUST support EVPN BGP routes (EVPN SAFI) and SHOULD support VPLS AD route (VPLS SAFI). All the logic for the integration will reside on the (PBB-)EVPN PEs side. However, if a VPLS instance is setup without the use of BGP auto-discovery, it is still possible (but cumbersome) for (PBB-)EVPN PEs to integrate into that VPLS instance.

3.1 Capability Discovery

The (PBB-)EVPN PEs must advertise both the BGP VPLS auto-discovery (AD) route as well as the BGP EVPN Inclusive Multicast route for a given VPN instance. The (PBB-)VPLS PEs only advertise the BGP VPLS AD route, per current standard procedures specified in [RFC4761] and [RFC6074]. The operator may decide to use the same BGP RT for both (PBB-)EVPN and (PBB-)VPLS. In this case, when a (PBB-)VPLS PE receives the EVPN Inclusive Multicast route, it will ignore it on the basis that it belongs to an unknown SAFI. However, the operator may use two RTs (one for (PBB-)VPLS and another for (PBB-)EVPN) and employ RT-constraint in order to prevent EVPN BGP routes from reaching the (PBB-)VPLS PEs. This provides an optimization in case required by the scale of the network.

When a (PBB-)EVPN PE receives both a VPLS AD route as well as an EVPN Inclusive Multicast route from a given remote PE for the same VPN instance, it MUST give preference to the EVPN route for the purpose of discovery. This ensures that, at the end of the route exchanges, all (PBB-)EVPN capable PEs discover other (PBB-)EVPN capable PEs as well as the (PBB-)VPLS-only PEs for that VPN instance. Furthermore, all the (PBB-)VPLS-only PEs would discover the (PBB-)EVPN PEs as if they were standard (PBB-)VPLS nodes. In other words, when the discovery phase is complete, the (PBB-)EVPN PEs would have discovered all the PEs in the VPN instance, and their associated capability: (PBB-)EVPN or VPLS-only. Whereas the (PBB-)VPLS PEs would have discovered all the PEs in the VPN instance, as if they were all VPLS-only nodes.

3.2 Forwarding Setup and Unicast Operation

The procedures for forwarding setup and unicast operation on the (PBB-)VPLS PE are per [RFC4447] and [PBB-VPLS].

The procedures for forwarding state setup and unicast operation on the (PBB-)EVPN PE are as follows:

- The (PBB-)EVPN PE must establish a pseudowire to a remote PE from which it has received only a VPLS AD route, for the VPN instance in question, and set up the label stack corresponding to the pseudowire FEC. This PW is between B-components of PBB-EVPN PE and PBB-VPLS PE per section 4 of [PBB-VPLS-PE-MODEL].

- The (PBB-)EVPN PE must set up the label stack corresponding to the MP2P (PBB-)VPN unicast FEC to any remote PE that has advertised EVPN AD route.

- If a (PBB-)EVPN PE receives a VPLS AD route followed by an EVPN AD route from the same PE and a pseudowire is setup to that PE, then the

(PBB-)EVPN MUST bring that pseudowire operationally down.

- If a (PBB-)EVPN PE receives an EVPN AD route followed by a VPLS AD route from the same PE, then the (PBB-)EVPN PE will setup the pseudowire but MUST keep it operationally down.

When the (PBB-)EVPN PE receives traffic over the pseudowires, it learns the associated MAC addresses in the data-plane. This is analogous to dynamic learning in IEEE bridges. If the PW belongs to the same split-horizon group as the EVPN mesh, then the MAC addresses learnt and associated to the PW will NOT be advertised in the control plane to any remote (PBB-)EVPN PE. The (PBB-)EVPN PE learns MAC addresses in the control plane, via the EVPN MAC Advertisement routes sent by remote (PBB-)EVPN PEs, and updates its MAC forwarding table accordingly. This is analogous to static learning in IEEE bridges. In PBB-EVPN, a given B-MAC address can be learnt either over the BGP control-plane from a remote PBB-EVPN PE, or in the data-plane over a pseudowire from a remote PBB-VPLS PE. There is no mobility associated with B-MAC addresses in this context. Hence, when the same B-MAC address shows up behind both a remote PBB-VPLS PE as well as a PBB-EVPN PE, the local PE can deduce that there is an anomaly in the network.

3.3 Multicast Operation

3.3.1 Ingress Replication The procedures for multicast operation on the (PBB-)VPLS PE, using ingress replication, are per [RFC4761], [RFC4762], and [PBB-VPLS].

The procedures for multicast operation on the PBB-EVPN PE, for ingress replication, are as follows:

- The PBB-EVPN PE builds a replication sub-list per I-SID to all the remote PBB-EVPN PEs in a given VPN instance, as a result of the exchange of the EVPN Inclusive multicast routes, as described in [PBB-EVPN]. This will be referred to as sub-list A. It comprises MP2P tunnels used for delivering PBB-EVPN BUM traffic [EVPN].

- The PBB-EVPN PE builds a replication sub-list per VPN instance to all the remote PBB-VPLS PEs , as a result of the exchange of the VPLS AD routes. This will be referred to as sub-list B. It comprises pseudowires from the PBB-EVPN PE in question to all the remote PBB-VPLS PEs in the same VPN instance.

- The PBB-EVPN PE may further prune sub-list B, on a per I-SID basis, if [MMRP] is run over the PBB-VPLS network. This will be referred to as sub-list C. This list comprises a pruned set of the pseudowires in sub-list B.

The replication list, maintained per I-SID, on a given PBB-EVPN PE will be the union of sub-list A and sub-list B if [MMRP] is NOT used, and the union of sub-list A and sub-list C if [MMRP] is used. Note that the PE must enable split-horizon over all the entries in the replication list, across both pseudowires and MP2P tunnels.

3.3.2 LSM Will be covered in a future revision of this document.

4 VPLS Integration with EVPN

4.1 Capability Discovery

The procedures for capability discovery are per Section 3.1 above.

4.2 Forwarding Setup and Unicast Operation

The operation here is largely similar to that of PBB-EVPN integration with PBB-VPLS, with the exception of the need to handle MAC mobility, the details of which will be covered in a future revision of this document.

4.3 Multicast Operation

4.3.1 Ingress Replication

The operation is per the procedures of Section 3.3.1 above for the scenario WITHOUT [MMRP]. The replication list is maintained per VPN instance, rather than per I-SID.

4.3.2 LSM Will be covered in a future revision of this document.

5 VPLS Integration with PBB-EVPN

5.1 Capability Discovery

The procedures for capability discovery are per Section 3.1 above.

5.2 Forwarding Setup and Unicast Operation

The operation here is largely similar to that of PBB-EVPN integration with PBB-VPLS, with a few exceptions listed below:

- When a PW is setup between a PBB-EVPN PE and a VPLS PE, it gets setup between the I-component of PBB-EVPN PE and the bridge component of VPLS PE.

- The MAC mobility needs to be handled. The details of which will be

covered in a future revision of this document.

5.3 Multicast Operation

5.3.1 Ingress Replication

The operation is per the procedures of Section 3.3.1 above for the scenario WITHOUT [MMRP]. The replication list is maintained per I-SID on the PBB-EVPN PEs and per VPN instance on the VPLS PEs.

5.3.2 LSM Will be covered in a future revision of this document.

6 Solution Advantages

The solution for seamless integration of (PBB-)EVPN with (PBB-)VPLS has the following advantages:

- When ingress replication is used for multi-destination traffic delivery, the solution reduces the scope of [MMRP] (which is a soft-state protocol) to only that of existing VPLS PEs, and uses the more robust BGP-based mechanism for multicast pruning among new EVPN PEs.

- It is completely backward compatible.

- New PEs can leverage the extensive multi-homing mechanisms and provisioning simplifications of PBB-EVPN:
   1. Auto-sensing of MHN / MHD
   2. Auto-discovery of redundancy group
   3.Auto-provisioning of DF election and VLAN carving

7 Security Considerations

No new security considerations beyond those for VPLS and EVPN.

8 IANA Considerations

This document has no actions for IANA.

9 References

9.1 Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate

Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4448] Martini, L., Ed., Rosen, E., El-Aawar, N., and G. Heron, "Encapsulation Methods for Transport of Ethernet over MPLS Networks", RFC 4448, April 2006.

[RFC4447] Martini, et al., "Pseudowire Setup and Maintenance using the Label Distribution Protocol", draft-ietf-pwe3-rfc4447bis-02.txt, October 2013.

[EVPN] Sajassi et al., "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-04.txt, work in progress, July, 2013.

[PBB-EVPN] Sajassi et al., "PBB-EVPN", draft-ietf-l2vpn-pbb-evpn-05.txt, work in progress, October, 2013.

9.2 Informative References

[MMRP] Clause 10 of "IEEE Standard for Local and metropolitan area networks - Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks", IEEE Std 802.1Q, 2013.

[PBB-VPLS-PE-MODEL] Balus, F., Sajassi, S., Bitar, N., "Extensions to VPLS PE model for Provider Backbone Bridging", RFC xxxx, June 2013.

[PBB-VPLS] Sajassi, et al., "VPLS Interoperability with Provider Backbone Bridges", draft-ietf-l2vpn-pbb-vpls-interop-05.txt, work in progress, October, 2013.

Authors' Addresses

Ali Sajassi
Cisco
Email: sajassi@cisco.com

Samer Salam
Cisco
595 Burrard Street, Suite 2123
Vancouver, BC V7X 1J1, Canada
Email: ssalam@cisco.com

Nick Del Regno
Verizon
400 International Pkwy
Richardson, TX 75089, US
Email: nick.delregno@verizon.com

Jorge Rabadan
Alcatel-Lucent
400 International Pkwy
Richardson, TX 75089, US
Email: jorge.rabadan@alcatel-lucent.com

Network Working Group S. Zhuang
Internet-Draft Z. Li
Intended status: Informational Huawei Technologies
Expires: February 18, 2015 August 17, 2014

Yang Model for Ethernet VPN
draft-zhuang-l2vpn-evpn-yang-cfg-00

Abstract

This document defines a YANG data model that can be used to configure and manage Ethernet VPN.

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].

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 February 18, 2015.

Copyright Notice

Table of Contents

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 intial 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 programatic APIs.

This document defines a YANG data model that can be used to configure and manage Ethernet VPN defined in [I-D.ietf-l2vpn-evpn].

2. Terminology

EVN: Ethernet Virtual Network

EVPN: Ethernet VPN

ESI: Ethernet Segment Identifier

3. Design of Data Model

3.1. Overview

The EVPN Yang module is divided in following containers :

o interfaces : that contains writable configuration objects of interface binded with EVPN.

o evpn : that contains global writable configuration objects of EVPN.

o evn-bgp: that contains writable configuration objects of MP-BGP used for EVPN.

o evn-instances : that contains writable configuration objects of EVPN instance.

The figure below describe the overall structure of the EVPN Yang module :

```
module: evn
   +--rw interfaces
   |  ...
   +--rw evn
   |  ...
   +--rw evn-bgp
   |  ...
   +--rw evn-instances
      ...
```

3.2. EVPN Interface Configuration

EVPN interface configuration includes the interface name and Ethernet Segment Identifier(ESI).

```
   +--rw interfaces
   |  +--rw interface* [name]
   |     +--rw name    leafref
   |     +--rw esi?    string
```

3.3. EVPN Global Configuration

EVPN global configuration includes the global parameters for ARP cache.

```
   +--rw evn
   |  +--rw arp-cache-disable?   boolean
   |  +--rw arp-cache-timeout?   uint32
```

3.4. MP-BGP Configuration for EVPN

The traditional configuration model of BGP is defined in [I-D.zhdankin-netmod-bgp-cfg]. In order to satifsy the requirment of reducing operation cost, this document proposes a new model of MP-BGP configuration for EVPN. A independent evn-bgp container is defined in EVPN Yang model to contains writable configuration objects of MP-BGP used for EVPN. It can directly configure MP-BGP peers for EVPN using the bgpPeers container. In addition, BGP router reflector can be introduced to reduced the configuration work for EVPN since when BGP router reflector is introduced each EVPN BGP client only needs to set up BGP peer with the router reflector. For BGP router reflector used for EVPN, it can enable the dynamic BGP peer setup mode to set up BGP peer with EVPN BGP client through the auto-discovery mechanism. Or it can adopt the traditional method to statically designate the list of EVPN BGP clients. The set-route-reflect-function container contains the writable configuration objects of BGP route reflector used for EVPN.

Besides above configuration, EVPN BGP configuation also includes the parameters of BFD and MAC limit.

```
      +--rw evn-bgp
      |  +--rw bfd
      |  |  +--rw isBfdEnable?   boolean
      |  |  +--rw txInterval?    uint32
      |  |  +--rw rxInterval?    uint32
      |  |  +--rw multiplier?    uint8
      |  +--rw mac-limit-per-peer
      |  |  +--rw mac-limit-value?          uint32
      |  |  +--rw mac-limit-alert-percent?   uint8
      |  |  +--rw (mac-limit-action)?
      |  |     +--:(enable-alert-only)
      |  |     |  +--rw alert-only?                boolean
      |  |     +--:(enable-idle-forever)
      |  |     |  +--rw idle-forever?              boolean
      |  |     +--:(enable-idle-timeout)
      |  |        +--rw idle-timeout?              uint16
      |  +--rw source-address?               inet:ip-address
      |  +--rw bgpPeers
      |  |  +--rw bgpPeer* [peerAddr]
      |  |     +--rw peerAddr    inet:ip-address
      |  +--rw set-route-reflect-function
      |     +--rw (set-type)?
      |     |  +--:(static)
      |     |  |  +--rw bgp-clients
      |     |  |     +--rw bgp-client* [clientAddr]
      |     |  |        +--rw clientAddr    inet:ip-address
      |     |  +--:(dynamic)
      |     |     +--rw server-enable?     boolean
      |     +--rw redundancy-mode?   enumeration
      |     +--rw df-delay-timer?    uint16
      |     +--rw timer
      |        +--rw keepaliveTime?   uint16
      |        +--rw holdTime?        uint16
```

3.5. EVPN Instance Configuration

EVPN instance configuration includes EVPN instance name, EVPN ID, and VLAN IDs in the VPN instance.

```
      +--rw evn-instances
         +--rw evn-instance* [evn-instance-name]
            +--rw evn-instance-name    string
            +--rw evn-id?              uint16
            +--rw vlan-ids
               +--rw vlan-id* [vlan-id-number]
                  +--rw vlan-id-number    uint16
```

4. EVPN Yang Module

EVN YANG MODEL
<CODE BEGINS> file "evn@2014-08-17.yang"

```
module evn {
  namespace "urn:huawei:params:xml:ns:yang:evn";
  // replace with IANA namespace when assigned
  prefix "evn";

  import ietf-interfaces {
    prefix if;
    //rfc7223-YANG Interface Management
  }

  import ietf-inet-types {
    prefix inet;
    //RFC6991
  }

  description
    "This YANG module defines the generic configuration data for
     EVN service.

     Terms and Acronyms

     EVN: Ethernet Virtual Network
     EVPN: Ethernet VPN
     ESI: Ethernet Segment Identifier

    ";

  revision 2014-08-17 {
    description
      "Initial revision.";
  }

  /*
   * ethernet segment ID config.
   */
  container interfaces {
    list interface {
      key "name";
      leaf name {
        type leafref {
          path "/if:interfaces/if:interface/if:name";
        }
      }
      leaf esi {
```

```
        description
          "Specify the ethernet segment ID.";

        config "true";
        type string {
          length "24";
          pattern "(^00([0-9a-fA-F]){2}\.(([0-9a-fA-F]){4}\.){3}
          (([0-9a-fA-F]){4})$)";
        }
      }
    }
  }

  /*
   * Enable Ethernet Virtual Network.
   */
  container evnGlobal {

    leaf evnEnable {
      config "true";
      type "boolean";
      default "false";
    }

    leaf arp-cache-disable {
      config "true";
      type boolean;
      default "false";
    }

    leaf arp-cache-timeout {
      config "true";
      type uint32 {
        range "0..100000";
      }
      default "240";
    }

  }
  /*
   * Configuring BFD for EVN BGP.
   */
  container evn-bgp {

    container bfd {
      leaf isBfdEnable {
        description "Enable BFD";
```

```
        config "true";
        type boolean;
        default "false";
      }

      leaf txInterval {
        description "Specify the minimum transmit interval";

        config "true";
        type uint32 {
          range "0..4294967295";
        }
      }
      leaf rxInterval {
        description "Specify the minimum receive interval";

        config "true";
        type uint32 {
          range "0..4294967295";
        }
      }
      leaf multiplier {
        description "Specify the detect multiplier";
        config "true";
        default "3";
        type uint8 {
          range "3..50";
        }
      }


    }

    container mac-limit-per-peer {

      leaf mac-limit-value {
        description
          "Specify Mac route limit value.";

        config "true";
        type uint32 {
          range "1..4294967295";
        }
      }
      leaf mac-limit-alert-percent {
        description
          "Specify maximum percentage value. Start to generate
           warning messages if it reaches maximum percentage
```

```
           value";

        config "true";
        type uint8 {
          range "1..100";
        }
        default "75";
      }

      choice mac-limit-type {

        case enable-alert-only {
          leaf alert-only {
            description
              "Allows the router to generate log message without
               terminating session when the maximum is exceeded.";

            config "true";
            type boolean;
            default "false";
          }
        }
        case enable-idle-forever {
          leaf idle-forever {
            description
              "Do not auto-connect-retry until reset bgp when the
               maximum is exceeded and then terminating session.";

            config "true";
            type boolean;
            default "false";
          }
        }
        case enable-idle-timeout {
          leaf idle-timeout {
            description
              "Specify Value of idle-timeout timer(minutes).
               Auto-connect-retry after timeout when the maximum is
               exceeded and then terminating session.";

            config "true";
            type uint16 {
              range "1..1200";
            }
          }
        }
      }
    }
```

```
    leaf source-address {
      config "true";
      type inet:ip-address;
    }

    /*
     * Configuring an Authentication Mode for EVN BGP.
     */
    container authentication {
      description
        "To improve network security, you can configure MD5 or
         Keychain authentication for EVN BGP peers when they
         set up a TCP connection.
        ";

      leaf cipherPassword {
        config "true";
        type "string";
      }
      leaf keychainName {
        config "true";
        type "string";
      }
    }

    container bgpPeers {
      list bgpPeer {
      key "peerAddr";
      max-elements "unbounded";
      min-elements "0";
        description
          "BGP Peer configure class.";

        leaf peerAddr {
          description
            "The nerighbor address.";
            config "true";
            type inet:ip-address;
            mandatory true;
        }

      }
    }


    container set-route-reflect-function {
      description
        "Configure an EVN BGP RR to reduce the number of EVN BGP
```

```
         peer connections, saving network resources.";

       choice set-type {
         description
           "An EVN BGP RR can be manually specified or dynamically
            configured.

            Static RR:
            After a static RR is configured, you need to manually
            establish peer relationships between the RR and other
            PE devices and specify the PE devices as the RR clients.

            Dynamic RR:
            Only a non-PE device can be configured as a dynamic RR.
            After a device is configured as a dynamic RR, it can
            automatically set up peer relationships with devices
            specified by the peer ip-address command (ip-address is
            the source address of the dynamic RR).
            ";

         case static {
           container bgp-clients {
             list bgp-client {
               key "clientAddr";
               max-elements "unbounded";
               min-elements "0";
               description
                 "Configure some peers as route reflector clients.";

               leaf clientAddr {
                 description
                   "The client address. A static RR is configured. Only
                    the specified peers can become the RR clients.";

                 config "true";
                 type inet:ip-address;
               }
             }
           }
         }

         case dynamic {
           leaf server-enable {
             description
               "Enable Server function for dynamic peer. A dynamic RR is
                configured. After a dynamic RR is configured, all PE
                devices that have established peer relationships with the
                RR can become the RR clients.";
```

```
            type boolean;
            default "false";
          }
        }

      }


      leaf redundancy-mode {
        description
          "Specify redundancy-mode.";

        config "true";
        type enumeration {
          enum "single-active";
          enum "all-active";
        }
        default "single-active";
      }


      leaf df-delay-timer{
        description
          "Specify designated forwarder election delay-timer
           value(seconds).";

        config "true";
        type uint32 {
          range "1..1200";
        }
        default "60";
      }

      container timer {
        leaf keepaliveTime {
          description "Specifies the Keepalive interval";
          config "true";
          default "60";
          type uint16 {
            range "0..21845";
          }
        }
        leaf holdTime {
          description "Specifies the Holdtime interval";
          config "true";
          default "180";
          type uint16 {
            range "0..65535";
```

```
          }
        }
      }


    }


  }


  container evnInstances {
    description
      "EVN instance configuration parameters.";

    list evnInstance {
      max-elements "unbounded";
      min-elements "0";
      key "evnName";

      leaf evnName {
        description
          "EVN Instance Name";

        config "true";
        type string {
          length "1..31";
        }
      }

      leaf evnId {
        description
          "Specify the EVN instance id. Each EVN instance has a unique
           ID.";

        config "true";
        type uint32 {
          range "1..65535";
        }
      }

      container vlanList {
        description
          "Specify a vlan list.";

        list vlan-id {
          key "vlan-id-number";
```

```
          leaf vlan-id-number {
            type uint16 {
              range "1..4094";
            }
          }
        }
      }

      container exportAclNameOrId {
        description
          "Filter outgoing routing updates. To accurately control EVN
           routes, configure an export routing policy. The export
           routing policy filters routes before they are advertised to
           other PE devices.";

        choice aclNumOrName {
          case Specify-aclNum {
            leaf aclNum {
              config "true";
              type uint16 {
                range "2000..2999";
              }
            }
          }
          case Specify-aclName {
            leaf aclName {
              config "true";
              type string;
            }
          }
        }
      }

      container importAclNameOrId {
        description
          "Set route filtering policy. To accurately control EVN routes,
           configure an import routing policy. The import routing policy
           filters routes received from other PE devices.";

        choice aclNumOrName {
          case Specify-aclNum {
            leaf aclNum {
              config "true";
              type uint16 {
                range "2000..2999";
              }
            }
          }
```

```
          case Specify-aclName {
            leaf aclName {
              config "true";
              type string;
            }
          }
        }
      }

      container evnInstanceInfo {
        description
          "Display the information of the evn instance.
           It is intended that this container may be augmented by
           vendors to reflect the vendor-specific operational state
           parameters.";

        leaf exportRT {
          config "false";
          type "string";
        }
        leaf importRT {
          config "false";
          type "string";
        }
        leaf evnRd {
          config "false";
          type "string";
        }
      }

    }

  }


}
</CODE ENDS>
```

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

This document does not introduce any new security risk.

7. Acknowledgements

The authors would like to thank Guangying Zheng, Gang Yan for their contributions to this work.

8. References

[I-D.ietf-l2vpn-evpn]
Sajassi, A., Aggarwal, R., Bitar, N., Isaac, A., and J. Uttaro, "BGP MPLS Based Ethernet VPN", draft-ietf-l2vpn-evpn-07 (work in progress), May 2014.

[I-D.zhdankin-netmod-bgp-cfg]
Alex, A., Patel, K., and A. Clemm, "Yang Data Model for BGP Protocol", draft-zhdankin-netmod-bgp-cfg-00 (work in progress), July 2013.

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

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

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

Authors' Addresses

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: February 25, 2015

S. Zhuang
H. Wang
Z. Li
Huawei Technologies
August 24, 2014

Yang Model for L2VPN
draft-zhuang-l2vpn-yang-cfg-00

Abstract

This document defines a YANG data model that can be used to configure and manage L2VPN. Both VPWS and VPLS are supported.

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].

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 February 25, 2015.

Copyright Notice

Table of Contents

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 L2VPN. Both VPWS and VPLS are supported.

2. Terminology

L2VPN: Layer 2 Virtual Private Network

VPLS: Virtual Private LAN Service

VPWS: Virtual Private Wire Service

3. Design of Data Model

3.1. Overview

The L2VPN Yang module is divided in following containers :

o l2vpncommon : that contains common writable configuration and readable objects for VPWS and VPLS.

o l2vpnvpws : that contains writable configuration and readable objects for VPWS.

o l2vpnvpls: that contains writable configuration and readable objects for VPLS.

The figure below describe the overall structure of the L2VPN Yang module :

```
module: l2vpn
   +--rw l2vpncommon
   ...
   +--rw l2vpnvpws
   ...
   +--rw l2vpnvpls
   ...
      ...
```

3.2. L2VPN Common Configuration

L2VPN common configuration container includes the global parameters for L2VPN, PW template configuration, etc. These parameters can be used by both VPWS and VPLS.

PW template configuration includes peer address, control word, MTU, sequence number, tunnel policy, parameters of AC, etc.

```
+--rw l2vpncommon
|  +--rw l2vpnGlobal
|  |  +--rw l2vpnEnable             boolean
|  |  +--rw vplsLoopDetectEnable?   boolean
|  +--rw pwTemplates
|  |  +--rw pwTemplate* [pwTemplateName]
|  |     +--rw pwTemplateName          string
|  |     +--rw peerAddr?               inet:ip-address
|  |     +--rw mtu?                    uint16
|  |     +--rw ctrlWord?               enumeration
|  |     +--rw tunnelPolicy?           string
|  |     +--rw tdmEncapsulateNumber?   uint8
|  |     +--rw jitterBuffer?           uint16
|  |     +--rw rtpHeader?              boolean
|  |     +--rw idleCode?               string
|  |     +--rw tdmSequenceNumber?      boolean
|  |     +--rw payloadCompression?     boolean
|  |     +--rw timeSlot?               uint8
|  |     +--rw maxAtmCells?            uint8
|  |     +--rw atmPackOvertime?        uint16
|  |     +--rw atmTransmitCell?        uint8
|  |     +--rw sequenceNumber?         boolean
...
```

3.3. VPWS Configuration

The VPWS configuration container includes VPWS instance configuration, VPWS switch instance configuration and VPWS statistics information.

```
+--rw l2vpnvpws
|  +--rw vpwsStatisticInfo
|  ...
|  +--rw vpwsInstances
|  ...
|  +--rw vpwsSwitchInstances
|  ...
```

3.3.1. VPWS Instances Configuration

The VPWS instance configuration includes per-instance parameters, AC configuration, PW configuration, TDM parameters, ATM parameters, reliability (including PW redundancy) configuration etc.

```
   +--rw vpwsInstances
   |  +--rw vpwsInstance* [instanceName instanceType]
   |     +--rw instanceName       leafref
   |     +--rw instanceType       instanceType
   |     +--rw encapsulateType?   pw-encapsulation
   |     +--rw description?       string
   |     +--ro instanceState?     enumeration
   |     +--ro lastUpTime?        yang:date-and-time
   |     +--ro totalUpTime?       string
   |     +--rw tdmParameters
   |     |  +--rw tdmEncapsulateNumber?   uint8
   |     |  ...
   |     +--rw atmParameters
   |     |  ...
   |     +--rw l2vpnAcs
   |     |  ...
   |     +--rw vpwsPws
   |     |  ...
   |     +--rw reliabilitys
   |        +--rw reliability* [pwRedundancyMode]
   ...
```

3.3.2. VPWS Switch Instances Configuration

VPWS switch instance configuration includes the configuration for multi-segment PW such as per-instance parameters, PW configuration, ATM parameters, TDM parameters etc.

```
   +--rw vpwsSwitchInstances
      +--rw vpwsSwitchInstance* [instanceName instanceType]
         +--rw instanceName       string
         +--rw instanceType       instanceType
         +--rw encapsulateType?   pw-encapsulation
         +--rw switchType?        enumeration
         +--rw ctrlWordTrans?     boolean
         +--rw controlWord?       enumeration
         +--ro instanceState?     enumeration
         +--ro createTime?        string
         +--ro upTime?            string
         +--ro lastChgTime?       string
         +--ro lastUpTime?        yang:date-and-time
         +--ro totalUpTime?       string
         +--rw vpwsPws
            +--rw vpwsPw* [pwRole pwId]
               +--rw pwRole            pw-role
               +--rw pwId              uint32
               +--rw peerIp?           inet:ip-address
               +--rw transmitLabel?    uint32
               +--rw receiveLabel?     uint32
               +--rw ctrlWord?         enumeration
               +--rw vccvAbility?      boolean
               +--rw tnlPolicyName?    string
               +--rw pwTemplateName?   string
               +--rw requestVlanId?    uint16
               +--rw vlanTpId?         string
               +--rw pwTtl?            uint8
               +--rw tdmParameters
               ...
               +--rw atmParameters
               ...
               +--rw vpwsLdpPwInfo
   ...
```

3.3.3. VPWS Statistics Information

The VPWS statistics information container includes statistics information of VPWS.

```
   +--rw vpwsStatisticInfo
   |  +--rw vpwsLdpAcStatInfo
   |  |  +--ro totalLdpAcNum?   uint32
   |  |  +--ro upLdpAcNum?      uint32
   |  |  +--ro downLdpAcNum?    uint32
   |  +--rw vpwsLdpPwStatInfo
   |  |  +--ro totalLdpPwNum?   uint32
   |  |  +--ro upLdpPwNum?      uint32
   |  |  +--ro downLdpPwNum?    uint32
   |  +--rw vpwsLdpPwRemoteStatInfo
   |  |  +--ro remoteVcNum?   uint32
   |  +--rw vpwsSwitchInstanceStatInfo
   |     +--ro totalSwitchInstanceNum?   uint32
   |     +--ro upSwitchInstanceNum?      uint32
   |     +--ro downSwitchInstanceNum?    uint32
```

3.4. VPLS Configuration

The L2VPN VPLS configuration includes VPLS instance configuration, VPLS statistics information.

```
   +--rw l2vpnvpls
      +--rw vplsStatisticInfo
      |  +--rw vplsInstStatisticsInfo
      ...
      |  +--rw vplsPwStatisticsInfo
      ...
      |  +--rw vplsAcStatisticsInfo
      ...
      |  +--ro vplsTnlRefInfos
      ...
      |  +--rw vplsLoopDetectStaticInfo
      |     +--ro totalVplsLoopDetectNum?   uint32
      +--rw vplsInstances
         +--rw vplsInstance* [instanceName]
            +--rw instanceName            string
            +--rw description?            string
            +--rw memberDiscoveryMode?    enumeration
            +--rw encapsulateType?        pw-encapsulation
            +--rw mtuValue?               uint16
            ...
```

3.4.1. VPLS Instance Configuration

The VPLS instance configuration includes member discovery mode, encapsulate type, VPLS LDP instance configuration, VPLS BGP AD instance configuration, VPLS BGP instance configuration and VPLS ACs configuration etc.

-- VPLS LDP instance configuration: This configuration describes how to configure LDP-based VPLS, with the signaling type being LDP.

-- VPLS BGP AD instance configuration: This configuration describes how to configure BGP AD VPLS to exchange extended BGP packets to automatically discover member VSIs in a VPLS domain and then use LDP FEC 129 to negotiate PW establishment to achieve automatic VPLS PW deployment.

-- VPLS BGP instance configuration: This configuration describes how to configure BGP VPLS. Detailed operations include configuring BGP as the signaling protocol, and configuring VPN targets to implement automatic discovery of VPLS PEs.

-- VPLS ACs configuration: This configuration describes configuration parameters of ACs.

```
+--rw vplsInstances
   +--rw vplsInstance* [instanceName]
      +--rw instanceName            string
      +--rw description?            string
      +--rw memberDiscoveryMode?    enumeration
      +--rw encapsulateType?        pw-encapsulation
      +--rw mtuValue?               uint16
      ...
      +--rw vsiPipe
      ...
      +--rw vplsLdpInst
      |  +--rw vsiId?               uint32
      ...
      +--rw vplsBgpAdInst
      |  +--rw vplsId?           string
      |  +--ro bgpAdRd?          string
      |  +--ro vsiId?            inet:ip-address
      |  +--rw vpnTargets
      ...
      +--rw vplsBgpInst
      |  +--rw bgpRd?          string
      |  +--rw ignoreMtu?      boolean
      ...
      +--rw vplsAcs
      |  +--rw vplsAc* [interfaceName]
      ...
      +--rw vplsLoopDetectInfo
         ...
```

3.4.2. VPLS Statistics Information

The VPLS statistics information container includes VPLS instance statistics information, VPLS PW statistics information, VPLS AC statistics information etc.

```
+--rw vplsStatisticInfo
|  +--rw vplsInstStatisticsInfo
...
|  +--rw vplsPwStatisticsInfo
...
|  +--rw vplsAcStatisticsInfo
...
|  +--ro vplsTnlRefInfos
...
|  +--rw vplsLoopDetectStaticInfo
|     +--ro totalVplsLoopDetectNum?   uint32
...
```

4. L2VPN Yang Module

<CODE BEGINS> file "l2vpn-yang@2014-08-21.yang"

```
module l2vpn {
    namespace "urn:huawei:params:xml:ns:yang:l2vpn";
    prefix "l2vpn";

    /* import */
    import ietf-inet-types {
        prefix inet;
    }
    import ietf-interfaces {
        prefix if;
    }
    import ietf-yang-types {
        prefix yang;
    }
    description
        "This YANG module defines the generic configuration data for
         L2VPN services.

         Terms and Acronyms
         L2VPN: Layer 2 Virtual Private Network
         VPLS: Virtual Private LAN Service
         VPWS: Virtual Private Wire Service
         ...
         ";
```

```
    revision 2014-08-21 {
        description
            "Initial revision.";
    }

    /* Typedef */
    typedef pw-encapsulation {
        description "PW encapsulation type.";
        type enumeration {
            enum fr {
                value "0";
                description "fr:";
            }
            enum atm-aal5-sdu {
                value "1";
                description "atm-aal5-sdu:";
            }
            enum atm-trans-cell {
                value "2";
                description "atm-trans-cell:";
            }
            enum vlan {
                value "3";
                description "vlan:";
            }
            enum ethernet {
                value "4";
                description "ethernet:";
            }
            enum hdlc {
                value "5";
                description "hdlc:";
            }
            enum ppp {
                value "6";
                description "ppp:";
            }
            enum cem {
                value "7";
                description "cem:";
            }
            enum atm-nto1-vcc {
                value "8";
                description "atm-nto1-vcc:";
            }
            enum atm-nto1-vpc {
                value "9";
                description "atm-nto1-vpc:";
```

```
            }
            enum ip-layer2 {
                value "10";
                description "ip-layer2:";
            }
            enum atm-1to1-vcc {
                value "11";
                description "atm-1to1-vcc:";
            }
            enum atm-1to1-vpc {
                value "12";
                description "atm-1to1-vpc:";
            }
            enum atm-aal5-pdu {
                value "13";
                description "atm-aal5-pdu:";
            }
            enum fr-port-mode {
                value "14";
                description "fr-port-mode:";
            }
            enum cesop {
                value "15";
                description "cesop:";
            }
            enum satop-e1 {
                value "16";
                description "satop-e1:";
            }
            enum satop-t1 {
                value "17";
                description "satop-t1:";
            }
            enum satop-e3 {
                value "18";
                description "satop-e3:";
            }
            enum satop-t3 {
                value "19";
                description "satop-t3:";
            }
            enum cesopsn-basic {
                value "20";
                description "cesopsn-basic:";
            }
            enum tdmoip_aal1 {
                value "21";
                description "tdmoip_aal1:";
```

```
            }
            enum cesopsn_tdm {
                value "22";
                description "cesopsn_tdm:";
            }
            enum tdmoip_aal2 {
                value "23";
                description "tdmoip_aal2:";
            }
            enum fr_dlci {
                value "24";
                description "fr_dlci:";
            }
            enum ip-interworking {
                value "25";
                description "ip-interworking:";
            }
            enum unsupport {
                value "26";
                description "unsupport:";
            }
        }
    }

    typedef pw-role {
        description "pw role.";
        type enumeration {
            enum primary {
                value "0";
                description "primary:";
            }
            enum backup {
                value "1";
                description "backup:";
            }
            enum bypass {
                value "2";
                description "bypass:";
            }
            enum multiHopOneSidePrimary {
                value "3";
                description "multiHopOneSidePrimary:";
            }
            enum multiHopOtherSidePrimary {
                value "4";
                description "multiHopOtherSidePrimary:";
            }
            enum multiHopOtherSideBackup {
```

```
                value "5";
                description "multiHopOtherSideBackup:";
            }
        }
    }

    typedef tunnelType {
        description "Indicates the type of tunnel used by the PW.";
        type enumeration {
            enum invalid {
                value "0";
                description "invalid tunnel type";
            }
            enum ldp {
                value "1";
                description "LDP LSP";
            }
            enum bgp {
                value "2";
                description "BGP LSP";
            }
            enum te {
                value "3";
                description "TE tunnel";
            }
            enum static_lsp {
                value "4";
                description "static lsp";
            }
            enum gre {
                value "5";
                description "GRE tunnel";
            }
            enum uni {
                value "6";
                description "uni tunnel";
            }
            enum tnl_group {
                value "7";
                description "tnl-group";
            }
            enum sub_te {
                value "8";
                description "TE sub tunnel";
            }
            enum sub_group {
                value "9";
                description "sub tunnel group";
```

```
            }
            enum 6over4 {
                value "10";
                description "manual IPv6 tunnel carry IPv4 traffic";
            }
            enum 6to4 {
                value "11";
                description "automatic IPv6 tunnel carry IPv4 traffic";
            }
            enum bgp_local_ifnet {
                value "12";
                description "BGP created mpls localifnet tunnel";
            }
            enum ldp6 {
                value "13";
                description "IPv6 LDP LSP";
            }
        }
    }

    typedef ifState {
        description "Interface state.";
        type enumeration {
            enum down {
                value "0";
                description "down:";
            }
            enum up {
                value "1";
                description "up:";
            }
            enum plugOut {
                value "2";
                description "plugOut:";
            }
            enum notifyDown {
                value "3";
                description "notifyDown:";
            }
            enum downNotify {
                value "4";
                description "downNotify:";
            }
        }
    }

    typedef pwState {
        description "Indicates the status of the PW.";
```

```
        type enumeration {
            enum down {
                value "0";
                description "down:";
            }
            enum up {
                value "1";
                description "up:";
            }
            enum backup {
                value "2";
                description "backup:";
            }
        }
    }

    typedef instanceType {
        description "Instance type. ";

        type enumeration {
            enum vpwsLocalccc {
                value "0";
                description "vpwsLocalccc:";
            }
            enum vpwsRemoteccc {
                value "1";
                description "vpwsRemoteccc:";
            }
            enum vpwsSvc {
                value "2";
                description "vpwsSvc:";
            }
            enum vpwsLdp {
                value "3";
                description "vpwsLdp:";
            }
            enum vpwsSwitch {
                value "4";
                description "vpwsSwitch:";
            }
            enum vpls {
                value "5";
                description "vpls:";
            }
        }
    }

    /* Grouping */
```

```
    grouping tdmParameter {
        description
            "Configure TDM parameter.";
        leaf tdmEncapsulateNumber {
            description "Number of encapsulated TDM frames.";
            config "true";
            type uint8 {
                range "1..40";
            }
        }
        leaf jitterBuffer {
            description "Depth of the TDM jitter buffer.";
            config "true";
            type uint16 {
                range "1000..64000";
            }
        }
        leaf rtpHeader {
            description
                "Whether or not the RTP header is added into the
                 transparently transported TDM frame.";
            config "true";
            type boolean;
        }
        leaf idleCode {
            description
                "Specifies the value of the idle code that is filled
                 manually when the jitter buffer underflow occurs.";
            config "true";
            type string {
                length "1..2";
                pattern "^((([1-9]|[a-f]|[A-F])([0-9]|[a-f]|[A-F])?)|
                         (0([0-9]|[a-f]|[A-F])?))$";
            }
        }
        leaf tdmSequenceNumber {
            description "Enable the seq-number option.";
            config "true";
            type boolean;
        }
        leaf payloadCompression {
            description
                "Specifies the dynamic bandwidth allocation for payload
                 compression.";
            config "true";
            type boolean;
        }
        leaf timeSlot {
```

```
            description
                "Specifies the time slot of the serial interface.";
            config "true";
            type uint8 {
                range "1..32";
            }
        }
    }

    grouping atmParameter {

        description "Configure ATM parameter.";

        leaf maxAtmCells {
            description "Maximum number of transmitted ATM cells.";
            config "true";
            type uint8 {
                range "1..28";
            }
        }
        leaf atmPackOvertime {
            description "Delay in packing ATM cells.";
            config "true";
            type uint16 {
                range "100..10000";
            }
        }
        leaf atmTransmitCell {
            description "ATM transmit cell.";
            config "true";
            type uint8 {
                range "1..28";
            }
        }
        leaf sequenceNumber {
            description
                "Enable the seq-number option.";
            config "true";
            type boolean;
        }
    }

    grouping speInfos {

        leaf speCount {
            description "Number of Spe.";
            config "false";
            type uint32;
```

```
        }

        list speInfo {

            config "false";
            key "spePwId spePeerIp";

            leaf spePwId {
                description
                    "Indicates the identifier of the PW.";
                type uint32;
            }
            leaf spePeerIp {
                description
                    "Specifies the LSR ID of the peer PE.";
                type inet:ip-address;
            }
        }
    }

    grouping vpwsPws {

        list vpwsPw {

            key "pwRole pwId";
            description "L2vpn vpws pw class.";

            leaf pwRole {
                description
                    "VPWS pw role:primary, backup,bypass.";
                config "true";
                type pw-role;

            }
            leaf pwId {
                description
                    "Indicates the identifier of the PW.";
                config "true";
                type uint32 {
                    range "1..4294967295";
                }
            }
            leaf peerIp {
                description
                    "Specifies the LSR ID of the peer PE.";
                config "true";
                type inet:ip-address;
            }
```

```
            leaf transmitLabel {
                description
                    "Indicates the label value of sent packets.";
                config "true";
                type uint32 {
                    range "0..1048575";
                }
            }
            leaf receiveLabel {
                description
                    "Indicates the label value of received packets.";
                config "true";
                type uint32 {
                    range "16..32767";
                }
            }
            leaf ctrlWord {
                description
                    "Enables the control word function. The control word
                    function is usually enabled on PWs with encapsulation
                    types being TDM, ATM or FR.

                    By default:
                    The control word function is enabled for TDM-, ATM-, or
                    Frame Relay-encapsulated PWs if PW profiles are not used.
                    If a PW profile is used, the control word function can
                    be enabled only after the control word is explicitly
                    specified. The control word function can be enabled for
                    PWs that use other types of encapsulation only after
                    the control word is explicitly specified.";
                config "true";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                    enum enable {
                        value "2";
                        description "enable:";
                    }
                }
            }
            leaf vccvAbility {
                description
```

```
                    "Configures VC connectivity detection. VC connectivity
                    detection supports two modes: control word mode and
                    label alert mode.";

                config "true";
                default "true";
                type boolean;
            }
            leaf tnlPolicyName {
                description
                    "Specifies a tunnel policy name for the L2VC. If no name
                    is specified for a tunnel policy, the default tunnel
                    policy is adopted. The LSP tunnel is preferred and only
                    one LSP is used for load balancing in the default tunnel
                    policy. If the name of the tunnel policy is specified but
                    no tunnel policy is configured, the default tunnel policy
                    is still adopted.";

                config "true";
                type string {
                    length "1..39";
                }
            }
            leaf pwTemplateName {
                description
                    "Specifies the name of a PW template. You can set
                     attributes for a PW template, including the remote
                     peer, tunnel policy, and control word. When configuring
                     an LDP PW, you can directly apply the PW template rather
                     than specifying attributes for the PW.";

                config "true";
                type string {
                    length "1..19";
                }
            }
            leaf requestVlanId {
                description
                    "Indicates the requested VLAN ID.";
                config "true";
                type uint16 {
                    range "1..4094";
                }
            }
            leaf vlanTpId {
                description "Indicates the TPID of requested VLAN ID.";
                config "true";
                type string {
```

```
                length "1..4";
                pattern "^([0-9]|[a-f]|[A-F]){0,4}$";
            }
        }
        leaf pwTtl {
            description "Specify the TTL for PW.";
            config "true";
            type uint8 {
                range "1..255";
            }
        }
        container tdmParameters {
            uses tdmParameter;
        }

        container atmParameters {
            uses atmParameter;
        }

        container vpwsLdpPwInfo {

            leaf interfaceName {
                description
                    "Indicates the type and number of the AC
                     interface.";
                config "false";
                type leafref {
                    path "/if:interfaces/if:interface/if:name";
                }
            }
            leaf ifState {
                description "Indicates  status of the AC interface.";
                config "false";
                type ifState;
            }
            leaf sessionState {
                description
                    "Indicates the status of the LDP session established
                     between both ends of the VC.";
                config "false";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum down {
                        value "1";
                        description "down:";
```

```
                }
                enum up {
                    value "2";
                    description "up:";
                }
            }
        }
        leaf integrativeAcState {
            description
                "The integrative status of the AC.";
            config "false";
            type ifState;
        }
        leaf acState {
            description
                "Indicates the status of the AC.";
            config "false";
            type ifState;
        }
        leaf pwState {
            description
                "Indicates the status of the PW.";
            config "false";
            type pwState;
        }
        leaf pwId {
            description
                "Indicates the identifier of the PW.";
            config "false";
            type uint32;
        }
        leaf encapType {
            description
                "Indicates the encapsulation type of the PW.";
            config "false";
            type pw-encapsulation ;

        }
        leaf destination {
            description
                "Indicates the LSR ID of the VC peer device.";
            config "false";
            type inet:ip-address;
        }
        leaf localGroupId {
            description "Indicates the local group ID.";
            config "false";
            type uint32;
```

```
                }
                leaf remoteGroupId {
                    description "Indicates the remote group ID.";
                    config "false";
                    type uint32;
                }
                leaf localVcLabel {
                    description "Indicates the local VC label.";
                    config "false";
                    type uint32;
                }
                leaf remoteVcLabel {
                    description "Indicates the remote VC label.";
                    config "false";
                    type uint32;
                }
                container tdmInfo {

                    description "TDM info";
                    config "false";

                    leaf localTdmEncapsulateNum {
                        description "Number of encapsulated TDM frames.";
                        config "false";
                        type uint8;
                    }
                    leaf remoteTdmEncapsulateNum {
                        description "Number of encapsulated TDM frames.";
                        config "false";
                        type uint8;
                    }
                    leaf jitterBuffer {
                        description "Depth of the TDM jitter buffer.";
                        config "false";
                        type uint8;
                    }
                    leaf idleCode {
                        description
                             "Indicates the idle code that is filled manually
                              when the jitter buffer underflow occurs.";
                        config "false";
                        type string {
                            length "0..3";
                        }
                    }
                    leaf localRtpHeaderEnable {
                        description
                             "Whether or not the RTP header is added into
```

```
                        the transparently transported TDM frame.";
                    config "false";
                    type boolean;
                }
                leaf remoteRtpHeaderEnable {
                    description
                        "Whether or not the RTP header is added into the
                         transparently transported TDM frame.";
                    config "false";
                    type boolean;
                }
                leaf localBitRate {
                    description "Indicates the bit-rate of the local VC.";
                    config "false";
                    type uint16;
                }
                leaf remoteBitRate {
                    description "Indicates the bit-rate of the remoteVC.";
                    config "false";
                    type uint16;
                }
            }
            container atmInfo {

                description "TDM info";
                config "false";

                leaf maxAtmCells {
                    description "Maximum number of transmitted ATM cells.";
                    config "false";
                    type uint8 {
                        range "1..28";
                    }
                }
                leaf remoteMaxAtmCells {
                    description "Maximum number of transmitted ATM cells.";
                    config "false";
                    type uint8 {
                        range "0..28";
                    }
                }
                leaf atmPackOvertime {
                    description "Delay in packing ATM cells.";
                    config "false";
                    type uint16 {
                        range "100..10000";
                    }
                }
```

```
                leaf atmTransmitCell {
                    description "ATM transmit cell.";
                    config "false";
                    type uint16 {
                        range "1..28";
                    }
                }
                leaf sequenceNumber {
                    description
                         "Enable the seq-number option.By default, the
                          seq-number option is disabled.";
                    config "false";
                    type boolean;
                }
            }
            leaf localFwdState {
                description
                    "Indicates the status of the local forwarding table.";
                config "false";
                type enumeration {
                    enum notForwarding {
                        value "0";
                        description "notForwarding:";
                    }
                    enum forwarding {
                        value "1";
                        description "forwarding:";
                    }
                }
            }
            leaf localStateCode {
                description
                    "Indicates the status code of the local PW:
                    0x0: indicates that the local PW functioning as the
                         master PW is in the Up state.
                    0x20: indicates that the local PW functioning as the
                          backup PW is in the Up state.
                    0x1: indicates that the PW functioning as the master
                         PW and is in the Down state.
                    0x21: indicates that the PW functioning as the backup
                         PW and is in the Down state.";
                config "false";
                type uint32;
            }
            leaf remoteFwdState {
                description
                    "Indicates the status of the remote forwarding
                     table.";
```

```
                    config "false";
                    type enumeration {
                        enum notForwarding {
                            value "0";
                            description "notForwarding:";
                        }
                        enum forwarding {
                            value "1";
                            description "forwarding:";
                        }
                    }
                }
                leaf remoteStateCode {
                    description
                        " Indicates the status code of the remote PW:
                        0x0: indicates that the remote PW functioning as the
                             master PW is in the Up state.
                        0x20: indicates that the remote PW functioning as the
                              backup PW is in the Up state.
                        0x1: indicates that the PW functioning as the master
                             PW and is in the Down state.
                        0x21: indicates that the PW functioning as the backup
                              PW and is in the Down state.";
                    config "false";
                    type uint32;
                }
                leaf isActive {
                    description
                        "Indicates whether or not the PW is in active state
                        (if so, user packets can be forwarded).";
                    config "false";
                    type boolean;
                }
                leaf isForwardExist {
                    description
                        "Indicates whether or not forwarding entries exist.";
                    config "false";
                    type boolean;
                }
                leaf linkState {
                    description
                        "Indicates the link status of the AC interface:
                        Up: indicates that the physical layer status of
                            the interface is functional.
                        Down: indicates that the physical layer of the
                              interface fails.";
                    config "false";
                    type enumeration {
```

```
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum down {
                        value "1";
                        description "down:";
                    }
                    enum up {
                        value "2";
                        description "up:";
                    }
                }
            }
            leaf localVcMtu {
                description "Indicates the MTU of the local VC.";
                config "false";
                type uint16;
            }
            leaf remoteVcMtu {
                description "Indicates the MTU of the remote VC.";
                config "false";
                type uint16;
            }
            leaf localVCCV {
                description
                    "Indicates the type of VCCV that is supported
                     locally. By default, the control word function
                     is not enabled, and the supported VCCV type is
                     alert lsp-ping bfd, indicating that LSP ping
                     and BFD are supported for the alert channel.
                     If the control word function is enabled, the
                     VCCV type is cw alert lsp-ping bfd, indicating
                     that LSP ping and BFD are supported both for the
                     control word channel and the alert channel.";
                config "false";
                type string {
                    length "0..40";
                }
            }
            leaf remoteVCCV {
                description
                    "Indicates the type of VCCV that is supported
                     remotely. By default, the control word function
                     is not enabled, and the supported VCCV type is
                     alert lsp-ping bfd, indicating that LSP ping and
                     BFD are supported for the alert channel. If the
                     control word function is enabled, the VCCV type
```

```
                        is cw alert lsp-ping bfd, indicating that LSP ping
                        and BFD are supported both for the control word
                        channel and the alert channel.";

                    config "false";
                    type string {
                        length "0..40";
                    }
                }
                leaf localCtrlWord {
                    description
                        "Indicates whether or not the control word is enabled
                        on the local end.";
                    config "false";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
                }
                leaf remoteCtrlWord {
                    description
                        "Indicates whether or not the control word is enabled
                         on the remote end.";
                    config "false";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
```

```
                }
                leaf tnlPolicyName {
                    description "Indicates the name of the tunnel policy.";
                    config "false";
                    type string {
                        length "1..39";
                    }
                }
                leaf pwTemplateName {
                    description "Indicates the name of the PW template.";
                    config "false";
                    type string {
                        length "1..19";
                    }
                }
                leaf priOrSec {
                    description
                        "Indicates whether the local status of the VC is
                         primary or secondary.";
                    config "false";
                    type enumeration {
                        enum primary {
                            value "0";
                            description "primary:";
                        }
                        enum secondary {
                            value "1";
                            description "secondary:";
                        }
                        enum bypass {
                            value "2";
                            description "bypass:";
                        }
                    }
                }
                leaf tunnelCount {
                    description
                        "Indicates that the PW uses one tunnel or token";
                    config "false";
                    type uint8;
                }
                container tunnelInfos {

                    config "false";
                    list tunnelInfo {

                        key "tunnelKey";
```

```
                        leaf tunnelKey {
                            description
                                "Indicates the ID of the tunnel used by the
                                 PW.";

                            type string {
                                length "0..21";
                            }
                        }
                        leaf tunnelType {
                            description
                                "Indicates the type of tunnel used by the
                                 PW.";
                            config "false";
                            type tunnelType;
                        }
                        leaf tunnelName {
                            description
                                "Indicates the name of the tunnel used by the
                                 PW.";
                            config "false";
                            type string {
                                length "0..64";
                            }
                        }
                        leaf publicNextHop {
                            description
                                "Indicates public next hop of a tunnel.";
                            config "false";
                            type inet:ip-address;
                        }
                    }

                }

                container speInfos {
                    config "false";
                    uses speInfos;
                }

                leaf createTime {
                    description
                        "Indicates how long the VC has been created for.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
```

```
                leaf upTime {
                    description
                        "Indicates how long the VC keeps the Up state.
                         If the PW is currently in the Down state, the
                         value is 0.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf lastChgTime {
                    description
                        "Indicates how long the VC status has remained
                         unchanged.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf pwLastUpTime {
                    description
                        "Indicates the last time the VC was Up.";
                    config "false";
                    type yang:date-and-time;
                }
                leaf pwTotalUpTime {
                    description
                        "Indicates the total duration the VC is Up.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }
                leaf supportNotification {
                    description
                        "Indicates whether or not the Notification
                         message is supported.";
                    config "false";
                    type boolean;
                }
            }

        }

    }

    grouping vplsPwInfo {
```

```
        leaf peerIp {
            description
                "Indicates the LSR ID of the VC peer device.";
            config "false";
            type inet:ip-address;
        }
        leaf pwId {
            description
                "Indicates the identifier of the PW.";
            config "false";
            type uint32;
        }
        leaf pwType {
            description "Type of the PW.label, QinQ,MEHVPLS";
            config "false";
            type enumeration {
                enum label {
                    value "0";
                    description "label:";
                }
                enum QinQ {
                    value "1";
                    description "QinQ:";
                }
                enum MEHVPLS {
                    value "2";
                    description "MEHVPLS:";
                }
            }
        }
        leaf sessionState {
            description
                "Indicates the status of the LDP session established
                 between both ends of the VC.";
            config "false";
            type enumeration {
                enum default {
                    value "0";
                    description "default:";
                }
                enum down {
                    value "1";
                    description "down:";
                }
                enum up {
                    value "2";
                    description "up:";
                }
```

```
            }
        }
        leaf pwState {
            description "Indicates the status of the PW.";
            config "false";
            type enumeration {
                enum down {
                    value "0";
                    description "down:";
                }
                enum up {
                    value "1";
                    description "up:";
                }
                enum backup {
                    value "2";
                    description "backup:";
                }
            }
        }
        leaf localVcLabel {
            description "Indicates the local VC label.";
            config "false";
            type uint32;
        }
        leaf remoteVcLabel {
            description "Indicates the remote VC label.";
            config "false";
            type uint32;
        }
        leaf tnlPolicyName {
            description
                "Indicates the name of the tunnel policy.";
            config "false";
            type string {
                length "1..39";
            }
        }
        leaf pwLastUpTime {
            description
                "Indicates the last time the VC was Up.";
            config "false";
            type yang:date-and-time;
        }
        leaf pwTotalUpTime {
            description
                "Indicates the total duration the VC is Up.";
            config "false";
```

```
            type string {
                length "1..80";
            }
        }
        container tunnelInfos {

            config "false";

            list tunnelInfo {

                key "tunnelKey";

                leaf tunnelKey {
                    description
                        "Indicates the ID of the tunnel used by the PW.";
                    type string {
                        length "0..21";
                    }
                }
                leaf tunnelType {
                    description
                        "Indicates the type of tunnel used by the PW.";
                    config "false";
                    type tunnelType;
                }
                leaf outIntf {
                    description
                        "Outbound interface.";
                    config "false";
                    type string {
                        length "0..256";
                    }
                }
                leaf tunnelName {
                    description
                        "Indicates the name of the tunnel used by the PW.";
                    config "false";
                    type string {
                        length "0..64";
                    }
                }
                leaf publicNextHop {
                    description "Assign public next hop of a tunnel.";
                    config "false";
                    type inet:ip-address;
                }
            }
```

```
        }
        container speInfos {
            config "false";
            uses speInfos;
        }
        leaf remoteGroupId {
            description "ID of the remote group.";
            config "false";
            type uint32;
        }
        leaf remoteMtu {
            description "Indicates the MTU of a remote VC.";
            config "false";
            type uint16;
        }
        leaf remoteVCCVcode {
            description "Indicates the VCCV of a remote VC.";
            config "false";
            type string {
                length "0..40";
            }
        }
        leaf remoteStateCode {
            description
                "Indicates the status of a remote VC, which can be:
                FORWARD: The remote VC is in the forwarding state.
                STANDBY: The remote VC is in the standby state.
                AC FAULT: The remote AC interface is faulty.
                PSN FAULT: The remote VC is faulty.
                NO FORWRD: The remote VC interface cannot forward packets
                           owing to other reasons. ";
            config "false";
            type enumeration {
                enum forward {
                    value "0";
                    description "forward:";
                }
                enum not-forward {
                    value "1";
                    description "not forward:";
                }
                enum standby {
                    value "2";
                    description "standby:";
                }
                enum ac-fault {
                    value "3";
                    description "ac fault:";
```

```
                }
                enum psn-fault {
                    value "4";
                    description "psn fault:";
                }
            }
        }
    }

    /* container */
    container l2vpncommon {

        container l2vpnGlobal {

            description "L2VPN golbal attribute.";

            leaf l2vpnEnable {
                description
                    "L2vpn enable flag.";
                type boolean;
                config "true";
                mandatory "true";
            }
            leaf vplsLoopDetectEnable {
                description
                    "Vpls mac withdraw loop detect enable flag.";
                type boolean;
                config "true";
            }
        }

        container pwTemplates {

            list pwTemplate {

                key "pwTemplateName";
                description "L2VPN pw template class.";

                leaf pwTemplateName {
                    description
                        "Specifies the PW template name. The value is a
                         case-sensitive string of 1 to 19 characters without
                         blank space.";
                    config "true";
                    type string {
                        length "1..19";
                    }
                }
```

```
                leaf peerAddr {
                    description
                        "Assign a peer IP address to a PW template.";
                    config "true";
                    type inet:ip-address;
                }
                leaf mtu {
                    description
                        "Configure the mtu value for PW template, 46 to 9600.";
                    config "true";
                    default "1500";
                    type uint16 {
                        range "46..9600";
                    }
                }
                leaf ctrlWord {
                    description
                        "Enable the control word in a PW template.";
                    config "true";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum disable {
                            value "1";
                            description "disable:";
                        }
                        enum enable {
                            value "2";
                            description "enable:";
                        }
                    }
                }
                leaf tunnelPolicy {
                    description
                        "Configure a tunnel policy for a PW template.";
                    config "true";
                    type string {
                        length "1..39";
                    }
                }
                uses tdmParameter;

                uses atmParameter;

            }
        }
```

```
        container notMatchRemoteLdpInfos {

            config "false";
            list notMatchRemoteLdpInfo {

                key "pwId peerIp encapsulateType";

                leaf pwId {
                    description
                        "After an ID is set for a VC, it cannot be changed.
                         Different VCs have different IDs.";
                    type uint32;
                }
                leaf peerIp {
                    description "Indicates the peer ip of the VC peer device.";
                    type inet:ip-address;
                }
                leaf encapsulateType{
                    description "Indicates the encapsualtion VC peer device.";
                    type pw-encapsulation;
                }
                leaf remoteLabel {
                    description "Indicates the remote VC label.";
                    config "false";
                    type uint32 ;
                }
                leaf remoteGroupId {
                    description "ID of the remote group.";
                    config "false";
                    type uint32;
                }
                leaf remoteMtu {
                    description "Indicates the MTU of a remote VC.";
                    config "false";
                    type uint16;
                }
                leaf remoteStateCode {
                    description
                        "Indicates the status of a remote VC, which can be:
                            FORWARD: The remote VC is in the forwarding state.
                            STANDBY: The remote VC is in the standby state.
                            AC FAULT: The remote AC interface is faulty.
                            PSN FAULT: The remote VC is faulty.
                            NO FORWRD: The remote VC interface cannot forward
                                       packets owing to other reasons.";
                    config "false";
                    type enumeration {
                        enum forward {
```

```
                            value "0";
                            description "forward:";
                        }
                        enum not-forward {
                            value "1";
                            description "not forward:";
                        }
                        enum standby {
                            value "2";
                            description "standby:";
                        }
                        enum ac-fault {
                            value "3";
                            description "ac fault:";
                        }
                        enum psn-fault {
                            value "4";
                            description "psn fault:";
                        }
                    }
                }
            }

        }
    }

    container l2vpnvpws {

        container vpwsStatisticInfo {

            container vpwsLdpAcStatInfo {

                leaf totalLdpAcNum {
                    description "Total number of L2VPN VPWS LDP AC.";
                    config "false";
                    type uint32;
                }
                leaf upLdpAcNum {
                    description "Up number of L2VPN VPWS LDP AC.";
                    config "false";
                    type uint32;
                }
                leaf downLdpAcNum {
                    description "Down number of L2VPN VPWS LDP AC.";
                    config "false";
                    type uint32;
                }
            }
```

```
            container vpwsLdpPwStatInfo {

                leaf totalLdpPwNum {
                    description
                        "Indicates the total number of established LDP PWs";
                    config "false";
                    type uint32;
                }
                leaf upLdpPwNum {
                    description "Number of LDP PWs in the up state.";
                    config "false";
                    type uint32;
                }
                leaf downLdpPwNum {
                    description "Number of LDP PWs in the down state.";
                    config "false";
                    type uint32;
                }
            }

            container vpwsLdpPwRemoteStatInfo {

                leaf remoteVcNum {
                    description
                        "Indicates the total number of created remote LDP
                         PWs.";
                    config "false";
                    type uint32;
                }
            }

            container vpwsSwitchInstanceStatInfo {

                leaf totalSwitchInstanceNum {
                    description
                        "Indicates the total number of established switch-vc";
                    config "false";
                    type uint32;
                }
                leaf upSwitchInstanceNum {
                    description "Number of switch-vc in the up state.";
                    config "false";
                    type uint32;
                }
                leaf downSwitchInstanceNum {
                    description "Number of switch-vc in the down state.";
                    config "false";
                    type uint32;
```

```
                }
            }

        }

        container vpwsInstances {

            list vpwsInstance {

                key "instanceName instanceType";

                description "L2vpn vpws instance class.";

                leaf instanceName {

                    description "Specifies VPWS instance name.";
                    config "true";

                    type leafref {
                        path "/if:interfaces/if:interface/if:name";
                    }
                }
                leaf instanceType {
                    description "VPWS instance type:ldp,localccc.";
                    config "true";
                    type instanceType;
                }
                leaf encapsulateType {
                    type pw-encapsulation;
                }
                leaf description {
                    description "Specifies a description for the VC.";
                    config "true";
                    type string {
                        length "1..64";
                    }
                }
                leaf instanceState {
                    description "VPWS instance state.";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
                            value "1";
                            description "up:";
```

```
                    }
                    enum adminDown {
                        value "2";
                        description "adminDown:";
                    }
                }
            }
            leaf lastUpTime {
                description
                    "Indicates how long the instance keeps the Up state.
                     If the PW is currently in the Down state, the value
                     is 0.";
                config "false";
                type yang:date-and-time;
            }
            leaf totalUpTime {
                description
                    "Indicates the total duration the instance is Up.";
                config "false";
                type string {
                    length "1..80";
                }
            }

            container tdmParameters {
                uses tdmParameter;
            }

            container atmParameters {
                uses atmParameter;
            }

            container l2vpnAcs {

                list l2vpnAc {

                    key "interfaceName";
                    description "L2VPN ac class.";

                    leaf interfaceName {
                        description "Specifies the AC interface name.";
                        config "true";
                        type leafref {
                            path "/if:interfaces/if:interface/if:name";
                        }
                    }
                    leaf state {
                        description "Indicates the status of the AC.";
```

```
            config "false";
            type enumeration {
                enum default {
                    value "0";
                    description "default:";
                }
                enum down {
                    value "1";
                    description "down:";
                }
                enum up {
                    value "2";
                    description "up:";
                }
            }
        }

        container l2vpnPipe {

            description "L2VPN pipe mode.";

            leaf pipeMode {
                description "Pipe mode.";
                default "uniform";
                type enumeration {
                    enum pipe {
                        value "0";
                        description "pipe:";
                    }
                    enum shortPipe {
                        value "1";
                        description "shortPipe:";
                    }
                    enum uniform {
                        value "2";
                        description "uniform:";
                    }
                }
            }
        }

        container ifLinkProtocolTran {

            description "lacp status";

            leaf protocolLacp {
                description "lacp status";
                config "true";
```

```
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                }
            }
            leaf protocolLldp {
                description "lldp status";
                config "true";
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                }
            }
            leaf protocolBpdu {
                description "bpdu status";
                config "true";
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                }
            }
            leaf protocolCdp {
                description "cdp status";
                config "true";
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
```

```
                                value "1";
                                description "disable:";
                            }
                        }
                    }
                    leaf protocolUdld {
                        description "udld status";
                        config "true";
                        type enumeration {
                            enum enable {
                                value "0";
                                description "enable:";
                            }
                            enum disable {
                                value "1";
                                description "disable:";
                            }
                        }
                    }
                }

            }

            container vpwsPws {
                uses vpwsPws;
            }

            container reliabilitys {

                list reliability {

                    key "pwRedundancyMode";

                    leaf pwRedundancyMode {
                        description
                            "Specifies the redundancy mode of VPWS
                             instance.";

                        config "true";
                        type enumeration {
                            enum frr {
                                value "0";
                                description "frr:";
                            }
                            enum masterSlave {
                                value "1";
```

```
                    description "masterSlave:";
                }
                enum independent {
                    value "2";
                    description "independent:";
                }
            }
        }
        leaf switchover {
            description
                "Specifies switches traffic from the primary
                 PW to the secondary PW.";
            config "true";
            type boolean;
        }
        leaf dualReceive {
            description
                "Specifies enables a UPE interface to receive
                 packets from both the primary and secondary
                 PWs.";
            config "true";
            type boolean;
        }
        container reRoute {

            description "L2vpn vpws pw reroute class.";

            leaf reRoutePolicy {
                description "Specifies the Policy of Reroute.";
                config "true";
                type enumeration {
                    enum delay {
                        value "0";
                        description "delay:";
                    }
                    enum immediately {
                        value "1";
                        description "immediately:";
                    }
                    enum never {
                        value "2";
                        description "never:";
                    }
                }
            }
            leaf delayTime {
                description
                    "Specifies the delay for switching traffic
```

```
                    back to the primary PW. ";
                config "true";
                type uint16 {
                    range "10..600";
                }
            }
            leaf resumeTime {
                description
                    "Specifies the time after which the peer PE
                     on the secondary PW is notified that the
                     local PE is recovered from the fault. ";
                config "true";
                type uint16 {
                    range "0..600";
                }
            }
            leaf lastReRouteReason {
                description
                    "Specifies the reason of Last Reroute.";
                config "false";
                type string {
                    length "0..100";
                }
            }
            leaf lastReRouteTime {
                description
                    "Specifies the time of Last Reroute.";
                config "false";
                type string {
                    length "1..60";
                }
            }
            leaf delayResidual {
                description
                    "Specifies the residual delay time for
                     switching traffic back to the primary PW.
                    ";
                config "false";
                type uint32;
            }
            leaf resumeResidual {
                description
                    "Specifies the residual time after which
                     the peer PE on the secondary PW is
                     notified that the local PE is recovered
                     from the fault. ";
                config "false";
                type uint32;
```

```
                        }
                    }

                }

            }

        }

        container vpwsSwitchInstances {

            list vpwsSwitchInstance {

                key "instanceName instanceType";

                description "L2vpn vpws instance class.";

                leaf instanceName {
                    description "Specifies VPWS instance name.";
                    config "true";
                    type string {
                        length "1..31";
                    }
                }
                leaf instanceType {
                    description "VPWS instance type:vpwsSwitch.";
                    config "true";
                    type instanceType;
                }
                leaf encapsulateType {
                    description "VPWS instance encapsulation type.";
                    config "true";
                    default "ethernet";
                    type pw-encapsulation;
                }
                leaf switchType {
                    description
                        "VPWS switch instance type:ldp2ldp,ldp2svc.";
                    config "true";
                    default "ldp2ldp";
                    type enumeration {
                        enum svc2svc {
                            value "0";
                            description "svc2svc:";
                        }
                        enum ldp2svc {
```

```
                        value "1";
                        description "ldp2svc:";
                    }
                    enum ldp2ldp {
                        value "2";
                        description "ldp2ldp:";
                    }
                    enum upe {
                        value "3";
                        description "upe:";
                    }
                }
            }
            leaf ctrlWordTrans {
                description
                    "Transparent transmission of control word If BFD
                     is enabled to monitor dynamic multi-hop PWs,
                     transparent transmission of control word needs
                     to be configured on the SPE. Otherwise, the BFD
                     negotiation fails. By default, transparent
                     transmission of control word is disabled.";
                config "true";
                type boolean;
                default "false";
            }
            leaf controlWord {
                description
                    "Enables the control word function. The control
                     word function is usually enabled on PWs with
                     encapsulation types being TDM, ATM or FR.
                     By default:
                     The control word function is enabled for TDM-,
                     ATM-, or Frame Relay-encapsulated PWs if PW
                     profiles are not used. If a PW profile
                     is used, the control word function can be
                     enabled only after the control word is explicitly
                     specified. The control word function can be enabled
                     for PWs that use other types of encapsulation only
                     after the control word is explicitly specified.";
                config "true";
                type enumeration {
                    enum default {
                        value "0";
                        description "default:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
```

```
                    }
                    enum enable {
                        value "2";
                        description "enable:";
                    }
                }
            }
            leaf instanceState {
                description "VPWS instance state.";
                config "false";
                type enumeration {
                    enum down {
                        value "0";
                        description "down:";
                    }
                    enum up {
                        value "1";
                        description "up:";
                    }
                    enum adminDown {
                        value "2";
                        description "adminDown:";
                    }
                }
            }
            leaf createTime {
                description
                    "Indicates how long the VC has been created for.";
                config "false";
                type string {
                    length "1..80";
                }
            }
            leaf upTime {
                description
                    "Indicates how long the VC keeps the Up state. If the
                     PW is currently in the Down state, the value is 0.";
                config "false";
                type string {
                    length "1..80";
                }
            }
            leaf lastChgTime {
                description
                    "Indicates how long the VC status has remained
                     unchanged.";
                config "false";
                type string {
```

```
                        length "1..80";
                    }
                }
                leaf lastUpTime {
                    description
                        "Indicates how long the instance keeps the Up state.
                         If the PW is currently in the Down state, the value
                         is 0.";
                    config "false";
                    type yang:date-and-time;
                }
                leaf totalUpTime {
                    description
                        "Indicates the total duration the instance is Up.";
                    config "false";
                    type string {
                        length "1..80";
                    }
                }

                container vpwsPws {
                    uses vpwsPws;
                }

            }

        }
    }

    container l2vpnvpls {

        container vplsStatisticInfo {

            container vplsInstStatisticsInfo {

                leaf totalVsiNum {
                    description "None";
                    config "false";
                    type uint32;
                }
                leaf vsiUpNum {
                    config "false";
                    type uint32;
                }
                leaf vsiDownNum {
                    config "false";
                    type uint32;
                }
```

```
                leaf ldpModeNum {
                    config "false";
                    type uint32;
                }
                leaf bgpVsiNum {
                    config "false";
                    type uint32;
                }
                leaf bgpAdVsiNum {
                    config "false";
                    type uint32;
                }
                leaf unspecifiedNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsPwStatisticsInfo {

                leaf totalPwNum {
                    description "None";
                    config "false";
                    type uint32;
                }
                leaf upPwNum {
                    config "false";
                    type uint32;
                }
                leaf downPwNum {
                    config "false";
                    type uint32;
                }
                leaf ldpPwNum {
                    config "false";
                    type uint32;
                }
                leaf bgpPwNum {
                    config "false";
                    type uint32;
                }
                leaf bgpAdPwNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsAcStatisticsInfo {
```

```
                leaf totalVplsAcNum {
                    config "false";
                    type uint32;
                }
                leaf upVplsAcNum {
                    config "false";
                    type uint32;
                }
                leaf downVplsAcNum {
                    config "false";
                    type uint32;
                }
            }

            container vplsTnlRefInfos {

                config "false";
                list vplsTnlRefInfo {

                    key "tnlPolicyName";

                    leaf tnlPolicyName {
                        description "None";
                        config "false";
                        type string {
                            length "1..39";
                        }
                    }
                    container tnlVsiRefInfos {

                        list tnlVsiRefInfo {

                             key "instanceName";

                             leaf instanceName {
                                config "false";
                                type string {
                                    length "1..31";
                                }
                            }
                        }

                    }

                }

            }
```

```
            container vplsLoopDetectStaticInfo {

                 leaf totalVplsLoopDetectNum {
                    config "false";
                    type uint32;
                }
            }

        }

        container vplsInstances {

            list vplsInstance {

                key "instanceName";

                leaf instanceName {
                    description
                        "Specifies VPLS instance name.";
                    config "true";
                    type string {
                        length "1..31";
                    }
                }
                leaf description {
                    description
                        "Specify the VPLS instance description.";
                    config "true";
                    type string {
                        length "1..64";
                    }
                }
                leaf memberDiscoveryMode {
                    description
                        "The VPLS member discovery mode for a created VSI.";
                    config "true";
                    default "default";
                    type enumeration {
                        enum default {
                            value "0";
                            description "default:";
                        }
                        enum auto {
                            value "1";
                            description "auto:";
                        }
                        enum static {
                            value "2";
```

```
                        description "static:";
                    }
                    enum bdmode {
                        value "3";
                        description "bd mode:";
                    }
                }
            }
            leaf encapsulateType {
                description "VPWS instance encapsulation type.";
                config "true";
                default "vlan";
                type pw-encapsulation;
            }
            leaf mtuValue {
                description
                    "MTU specified in dynamic PW signaling negotiation.";
                config "true";
                default "1500";
                type uint16 {
                    range "328..65535";
                }
            }
            leaf tnlPolicyName {
                description
                    "Specifies a tunnel policy name for the VSI. If no
                     name is specified for a tunnel policy, the default
                     tunnel policy is adopted. The LSP tunnel is
                     preferred and only one LSP is used for load balancing
                     in the default tunnel policy. If the name of the
                     tunnel policy is specified but no tunnel policy is
                     configured, the default tunnel policy is still adopted.
                     ";
                config "true";
                type string {
                    length "1..39";
                }
            }
            leaf shutdown {
                description
                    "Sometimes, because of service debugging or service
                     halt, a VSI can be disabled for function
                     modification.";
                config "true";
                default "false";
                type boolean;
            }
            leaf isolateSpoken {
```

```
                    description
                        "The isolate spoken command enables forwarding
                         isolation between AC interfaces, between UPE
                         PWs, and between ACs and UPE PWs on a VSI. The
                         undo isolate spoken command disables forwarding
                         isolation";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf unknownUnicastAction {
                    description
                        "Specifies the processing mode for received unknown
                         unicast frames.";
                    config "true";
                    type enumeration {
                        enum broadcast {
                            value "0";
                            description "broadcast:";
                        }
                        enum drop {
                            value "1";
                            description "drop:";
                        }
                        enum local-handle {
                            value "2";
                            description "local-handle:";
                        }
                        enum drop-learn {
                            value "3";
                            description "drop-learn:";
                        }
                    }
                }
                leaf macLearnEnable {
                    description
                        "Enables MAC address learning on a VSI.";
                    config "true";
                    default "true";
                    type boolean;
                }
                leaf macLearnStyle {
                    description
                        "Sets the MAC address learning style of a VSI.By
                         default, MAC address learning style is unqualify.
                         Currently, the VRP supports only the unqualified
                         mode.";
```

```
                    config "true";
                    default "unqualify";
                    type enumeration {
                        enum qualify {
                            value "0";
                            description "qualify:";
                        }
                        enum unqualify {
                            value "1";
                            description "unqualify:";
                        }
                    }
                }
                leaf macAgeTimer {
                    description
                        "Sets the aging time of MAC address entries in a VSI.
                         By default, the aging time of MAC address entries in
                         a VSI is the global aging time. You can set the global
                         aging time by the command mac-address aging-time
                         (system view).";
                    config "true";
                    type uint32 {
                        range "0..1000000";
                    }
                }
                leaf totalAcService {
                    description
                        "Total number of interface in the instance.";
                    config "false";
                    type uint32;
                }
                leaf createTime {
                    description
                        "Indicates how long the VSI has been created for.";
                    config "false";
                    type string {
                        length "1..60";
                    }
                }
                leaf vsiState {
                    description "VPLS instance state.";
                    config "false";
                    type enumeration {
                        enum down {
                            value "0";
                            description "down:";
                        }
                        enum up {
```

```
                        value "1";
                        description "up:";
                    }
                    enum adminDown {
                        value "2";
                        description "adminDown:";
                    }
                }
            }
            leaf ignoreAcStateEffect {
                description
                    "After the ignore-ac-state command is configured,
                     the VSI status is not subject to changes in the
                     status of the AC. That is, a VSI can go Up even
                     if no AC is connected to the VSI.";

                config "false";
                default "false";
                type boolean;
            }

            container vsiPipe {

                leaf pipeMode {
                    description "Pipe mode";
                    config "true";
                    default "uniform";
                    type enumeration {
                        enum pipe {
                            value "0";
                            description "pipe:";
                        }
                        enum shortPipe {
                            value "1";
                            description "shortPipe:";
                        }
                        enum uniform {
                            value "2";
                            description "uniform:";
                        }
                    }
                }
                leaf serviceClass {
                    description "service class";
                    config "true";
                    default "be";
                    type enumeration {
                        enum be {
```

```
                        value "0";
                        description "be:";
                    }
                    enum af1 {
                        value "1";
                        description "af1:";
                    }
                    enum af2 {
                        value "2";
                        description "af2:";
                    }
                    enum af3 {
                        value "3";
                        description "af3:";
                    }
                    enum af4 {
                        value "4";
                        description "af4:";
                    }
                    enum ef {
                        value "5";
                        description "ef:";
                    }
                    enum cs6 {
                        value "6";
                        description "cs6:";
                    }
                    enum cs7 {
                        value "7";
                        description "cs7:";
                    }
                }
            }
            leaf color {
                description "service color";
                config "true";
                default "green";
                type enumeration {
                    enum green {
                        value "0";
                        description "green:";
                    }
                    enum yellow {
                        value "1";
                        description "yellow:";
                    }
                    enum red {
                        value "2";
```

```
                            description "red:";
                        }
                    }
                }
                leaf dsName {
                    description "domain name";
                    config "true";
                    type string {
                        length "1..31";
                    }
                }
            }

            container vplsLdpInst {

                leaf vsiId {
                    description
                        "After an ID is set for a VSI, it cannot be
                         changed. Different VSIs have different IDs.";
                    config "true";
                    type uint32 {
                        range "1..4294967295";
                    }
                }
                leaf macWithdraw {
                    description
                        "Configures a VSI to delete the local MAC
                         addresses and informs all the remote peers
                         of the deletion when an AC fault or a UPE
                         fault occurs and the VSI remains Up.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf ifChgWithdraw {
                    description
                        "Configures PEs to send LDP MAC Withdraw
                         messages to all peers when the status of
                         the AC interface bound to the VSI changes.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf upeUpeMacWithdraw {
                    description
                        "Configures an NPE to forward the LDP MAC
                         Withdraw messages received from a UPE to
                         other UPEs.";
```

```
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf upeNpeMacWithdraw {
                    description
                        "Configures an NPE to forward the LDP MAC
                         Withdraw messages received from UPEs to
                         other NPEs.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf npeUpeMacWithdraw {
                    description
                        "Configures an NPE to forward the LDP MAC
                         Withdraw messages received from other NPEs
                         to UPEs.";
                    config "true";
                    type boolean;
                }
                container vplsLdpPws {

                    list vplsLdpPw {

                        key "peerIp pwId pwEncapType";

                        leaf peerIp {
                            description "Specifies the LSR ID of the peer PE
."";
                            config "true";
                            type inet:ip-address;
                        }
                        leaf pwId {
                            description
                                "Indicates the identifier of the PW. Default
,
                                 we may use vsiId as the pwId. Sometimes we
may
                                 create pw to different pw that the pwId not
                                 match our vsiId, so it must specify the pwI
d.";

                            config "true";
                            type uint32 {
                                range "1..4294967295";
                            }
                        }
                        leaf pwEncapType {
                            description
                                "Indicates the encapsulation of the PW.
                                 Default, we may use encapsulateType as
```

```
                    the pwEncapType. Sometimes we may create
                    pw that the pwEncapType not match our
                    encapsulateType, so it must specify the
                    pwEncapType.";
                config "true";
                type pw-encapsulation;
            }
            leaf pwRole {
                description
                    "VPLS pw role:primary, secondary.";
                config "true";
                default "primary";
                type pw-role;
            }
            leaf pwName {
                description
                    "Specifies the name of a PW, which is used
                     to distinguish the PW from other PWs. The
                     PW name must be unique in the same VSI,
                     but can be the same as the PW names in
                     other VSIs. ";
                config "true";
                type string {
                    length "1..15";
                }
            }
            leaf ifParaVCCV {
                description
                    "Deletes the VCCV byte (an interface
                     parameter) in the Mapping packet.";
                config "true";
                default "true";
                type boolean;
            }
            leaf isUpe {
                description "set VPLS PW as upe.";
                config "true";
                default "false";
                type boolean;
            }
            leaf tnlPolicyName {
                description
                    "Specifies a tunnel policy name for the
                     VPLS PW. If no name is specified for a
                     tunnel policy, the default tunnel policy
                     is adopted. The LSP tunnel is preferred
                     and only one LSP is used for load
                     balancing in the default tunnel policy.
```

```
                                    If the name of the tunnel policy is
                                    specified but no tunnel policy is
                                    configured, the default tunnel policy is
                                    still adopted.";
                                config "true";
                                type string {
                                    length "1..39";
                                }
                            }

                            container vplsLdpPwInfo {
                                config "false";
                                uses vplsPwInfo;
                            }
                        }

                    }
                }

                container vplsBgpAdInst {

                    leaf vplsId {
                        description
                            "Specifies the vpls id. The value is a
                             case-sensitive string of 3 to 21 characters
                             without blank space.";
                        config "true";
                        type string {
                            length "1..21";
                        }
                    }

                    leaf bgpAdRd {
                        description
                            "Specifies the Route Distinguisher. The value is
                             a case-sensitive string of 3 to 21 characters
                             without blank space.";
                        config "false";
                        type string {
                            length "1..21";
                        }
                    }

                    leaf vsiId {
                        description
                            "Specifies the negotiation ip address of the
                             local PE.";
                        config "false";
```

```
                    type inet:ip-address;
                }

                container vpnTargets {
                    description "BGP-AD vpn-targets.";
                    list vpnTarget {
                        key "vpnRTValue";
                        description "BGP AD vpn targets.";

                        leaf vpnRTValue {

                          description
                            "Vpn-target:
                             adds VPN target extended community attribute
                             to the export or import VPN target extended
                             community list. The vpn-target can be
                             expressed in either of the following formats:
                             (1)16-bit AS number:32-bit user-defined number
                                For example, 1:3. The AS number ranges from
                                0 to 65535. The user-defined number ranges
                                from 0 to 4294967295. The AS number and the
                                user-defined number cannot be 0s at the same
                                time. That is, a VPN target cannot be 0:0.
                             (2)32-bit IP address:16-bit user-defined number
                                For example, 192.168.122.15:1. The IP addres
s
                                ranges from 0.0.0.0 to 255.255.255.255. The
                                user-defined number ranges from 0 to 65535."
;

                          mandatory "true";
                          type string {
                            length "3..21";
                          }
                        }

                        leaf vrfRTType {
                          description
                            "Specifies the vpn target type,
                             export-extcommunity: specifies the extended
                             community attributes carried in routing
                             information to be sent. import-extcommunity:
                             receives routing information carrying
                             specified extended community attributes.";
                           mandatory "true";
                           type enumeration {
                           enum export_extcommunity {
                             value "0";
                             description "export-extcommunity:";
                           }
```

```
                        enum import_extcommunity {
                          value "1";
                          description "import-extcommunity:";
                        }
                        enum both {
                          value "2";
                          description
                            "export-extcommunity &
                             import-extcommunity:";
                        }
                      }
                    }
                  }
                }

                container bgpAdPeerInfos {
                    config "false";

                    list bgpAdPeerInfo {

                        key "peerRouterID";
                        leaf peerRouterID {
                            description
                                "The Router ID of the remote router.";
                            type inet:ip-address;
                        }

                        leaf vplsId {
                            description
                                "The vpls id. The value is a case-sensitive
                                 string of 3 to 21 characters without blank
                                 space.";
                            type string {
                                length "1..21";
                            }
                        }

                        leaf sourceAII {
                            description
                                "The source AII of the remote PE.";
                            type inet:ip-address;
                        }

                        leaf targetAII {
                            description
                                "The target AII of the remote PE.";
                            type inet:ip-address;
                        }
```

```
                        leaf peerType {
                          description
                            "Specifies the peer type.Static,Dynamic.";

                          type enumeration {
                                enum static {
                                  value "0";
                                  description "Static pw";
                                }
                                enum dynamic {
                                  value "1";
                                  description "Dynamic pw";
                                }
                          }
                        }

                        container bgpAdPwInfo {
                            config "false";

                            uses vplsPwInfo;
                        }

                    }
                }

            }

            container vplsBgpInst {

                leaf bgpRd {
                    description
                        "Specifies the Route Distinguisher. The value is a
                        case-sensitive string of 3 to 21 characters without
                        blank space.";
                    type string {
                        length "1..21";
                    }
                }

                leaf ignoreMtu {
                    description
                        "Ignore the mtu when negotiate pw.";
                    type boolean;
                }

                container vpnTargets {
                    description "BGP vpn-targets";
                    list vpnTarget {
```

```
        key "vpnRTValue";
        description
            "BGP vpn targets";

        leaf vpnRTValue {

          description
            "Vpn-target: adds VPN target extended community
             attribute to the export or import VPN target
             extended community list. The vpn-target can be
             expressed in either of the following formats:
            (1)16-bit AS number:32-bit user-defined number
               For example, 1:3. The AS number ranges from
               0 to 65535. The user-defined number ranges
               from 0 to 4294967295. The AS number and the
               user-defined number cannot be 0s at the same
               time. That is, a VPN target cannot be 0:0.
            (2)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. The
               user-defined number ranges from 0 to 65535.";

          mandatory "true";
          type string {
            length "3..21";
          }
        }

        leaf vrfRTType {
          description
            "Specifies the vpn target type,
             export-extcommunity: specifies the extended
             community attributes carried in routing
             information to be sent.
             import-extcommunity: receives routing
             information carrying specified extended
             community attributes.";

          mandatory "true";
          type enumeration {
                enum export_extcommunity {
                  value "0";
                  description "export-extcommunity:";
                }
                enum import_extcommunity {
                  value "1";
                  description "import-extcommunity:";
                }
```

```
                                enum both {
                                  value "2";
                                  description "export-extcommunity &
                                               import-extcommunity:";
                                }
                              }
                            }
                        }
                    }

                    container bgpSite {

                        leaf siteId {
                            description
                                "Specifies the ID of the site.";
                            mandatory "true";
                            type uint16 {
                                range "1..65535";
                            }
                        }
                        leaf siteRange {
                            description
                                "Specifies the ID of the site range.";
                            type uint16 {
                                range "1..65535";
                            }
                        }
                        leaf defaultOffset {
                            description
                                "Specifies the default offset of the site ID.";
                            type uint8 {
                                range "0..1";
                            }
                        }
                    }

                    container bgpPeerInfos {
                        config "false";

                        list bgpPeerInfo {

                            key "siteId";

                            leaf siteId {
                                description
                                    "The site id of the peer.";
                                type uint16 {
                                    range "1..65535";
```

```
                        }
                    }

                    container bgpPwInfo {
                        config "false";

                        uses vplsPwInfo;
                    }

                }
            }

        }

        container vplsAcs {

            list vplsAc {

                key "interfaceName";

                leaf interfaceName {
                    description "Specifies the AC interface name. ";
                    config "true";
                    type leafref {
                        path "/if:interfaces/if:interface/if:name";
                    }
                }
                leaf hubModeEnable {
                    description
                        "Change the VSI attribute of the local
                         interface from spoke to hub.By default,
                         the AC side of a VSI has the attribute
                         of spoke, and the PW side of a VSI has
                         the attribute of hub.";
                    config "true";
                    default "false";
                    type boolean;
                }
                leaf state {
                    description
                        "Indicates the status of the AC.";
                    config "false";
                    type ifState;
                }
                leaf lastUpTime {
                    description
                        "Indicates how long the AC keeps the Up state.
                         If the AC is currently in the Down state, the
```

```
                value is 0.";
            config "false";
            type yang:date-and-time;
        }
        leaf totalUpTime {
            description
                "Indicates the total duration the AC is Up.";
            config "false";
            type string {
                length "1..60";
            }
        }
        container ifLinkProtocolTran {

            description "lacp status";

            leaf protocolLacp {
                description "lacp status";
                config "true";
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                }
            }
            leaf protocolLldp {
                description "lldp status";
                config "true";
                type enumeration {
                    enum enable {
                        value "0";
                        description "enable:";
                    }
                    enum disable {
                        value "1";
                        description "disable:";
                    }
                }
            }
            leaf protocolBpdu {
                description "bpdu status";
                config "true";
                type enumeration {
```

```
                                enum enable {
                                    value "0";
                                    description "enable:";
                                }
                                enum disable {
                                    value "1";
                                    description "disable:";
                                }
                            }
                        }
                        leaf protocolCdp {
                            description "cdp status";
                            config "true";
                            type enumeration {
                                enum enable {
                                    value "0";
                                    description "enable:";
                                }
                                enum disable {
                                    value "1";
                                    description "disable:";
                                }
                            }
                        }
                        leaf protocolUdld {
                            description "udld status";
                            config "true";
                            type enumeration {
                                enum enable {
                                    value "0";
                                    description "enable:";
                                }
                                enum disable {
                                    value "1";
                                    description "disable:";
                                }
                            }
                        }
                    }
                }

                container vplsLoopDetectInfo {

                    description "L2vpn admin vsi binding class.";

                    leaf lastLoopType {
```

```
                        description
                            "Indicates the last mac withdraw loop type.";
                        config "false";
                        type enumeration {
                            enum detect-loop {
                                value "0";
                                description "detect loop:";
                            }
                            enum exceed-max-hop {
                                value "1";
                                description "exceed max hop:";
                            }
                        }
                    }
                    leaf sendPeer {
                        description
                            "Indicates the mac withdraw send peer.";
                        config "false";
                        type inet:ip-address;
                    }
                    leaf receivePeer {
                        description
                            "Indicates the mac withdrawreceive peer.";
                        config "false";
                        type inet:ip-address;
                    }
                    leaf lastLoopTime {
                        description
                            "Indicates the last mac withdraw loop time.";
                        config "false";
                        type string {
                            length "1..80";
                        }
                    }
                }

            }

        }

    }
}
</CODE ENDS>
```

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

This document does not introduce any new security risk.

7. Acknowledgements

The authors would like to thank Guangying Zheng, Gang Yan for their contributions to this work.

8. References

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

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

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

Authors' Addresses

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Haibo Wang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: rainsword.wang@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com

Network Working Group S. Zhuang
Internet-Draft Z. Li
Intended status: Informational Huawei Technologies
Expires: February 16, 2015 August 15, 2014

Yang Data Model for BGP/MPLS IP VPNs
draft-zhuang-l3vpn-yang-cfg-00

Abstract

This document defines a YANG data model that can be used to configure and manage L3VPN (BGP/MPLS IP VPN).

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].

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 February 16, 2015.

Copyright Notice

Table of Contents

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 encodings other than XML (e.g. JSON) are being defined. Furthermore, YANG data models can be used as the basis of implementation for other interfaces, such as CLI and programmatic APIs.

This document defines a YANG data model that can be used to configure and manage L3VPN (BGP/MPLS IP VPN) [RFC4364].

2. Definitions and Acronyms

AF: Address Family

BGP: Border Gateway Protocol

JSON: JavaScript Object Notation

L3VPN: Layer 3 VPN

NETCONF: Network Configuration Protocol

ReST: Representational State Transfer, a style of stateless interface and protocol that is generally carried over HTTP

YANG: A data definition language for NETCONF

3. Design of the L3VPN Model

3.1. Overview

The L3VPN Yang module consists of the following components :

o vpn-instances configuration : that contains per-instance writable configuration objects. VPN instances support both the IPv4 and IPv6 address families.

o vpn-interfaces configuration: that contains writable configuration objects of MPLS VPN interface.

o mp-bgp configuration: This component uses the definitions defined in BGP YANG module and augments some parameters.

o bgp-vpn-instance configuration: that contains writable configuration objects when using BGP between PE and CE.

The figure below describe the overall structure of the L3VPN Yang module :

```
module: l3vpn
   +--rw vpn-instances
   |  +--rw vpn-instance* [vpn-instance-name]
   |     +--rw vpn-instance-name    string
   |     +--rw description?         string
   |     +--rw ipv4-family
   |     |  +--rw route-distinguisher?   string
   |     |  +--rw vpnTargets
   |     |  |  +--rw vpnTarget* [vrfRTValue]
   |     |  |     +--rw vrfRTValue    string
   |     |  |     +--rw vrfRTType     enumeration
   | ...
   |     +--rw ipv6-family
   | ...
   +--rw vpn-interfaces
   |  +--rw vpn-interface* [name]
   |     +--rw name                 leafref
   |     +--rw vpn-instance-name?   string
   +--rw vrfInfo
      +--ro vrfCreateTime?   yang:timestamp
      ...
augment /bgp:bgp-router/bgp:vpnv4/bgp:unicast:
   +--rw apply-label-per-nexthop?   boolean
   +--rw upeEnable?                 boolean
augment /bgp:bgp-router/bgp:vpnv6/bgp:unicast:
   +--rw apply-label-per-nexthop?   boolean
   +--rw upeEnable?                 boolean
augment /bgp:bgp-router:
   +--rw bgp-af-ipv4-vpn-instances
   |  +--rw bgp-af-ipv4-vpn-instance* [vpn-instance-name]
   |...
   +--rw bgp-af-ipv6-vpn-instances
      +--rw bgp-af-ipv6-vpn-instance* [vpn-instance-name]
         +--rw vpn-instance-name    string
         +--rw router-id
         |...
```

3.2. VPN Instance Configuration

An instance is created to comprise the VPN forwarding information for each VPN in a BGP/MPLS IP VPN. This instance is called a VPN instance or a VPN routing and forwarding (VRF) table. It is also called a per-site forwarding table in [RFC4364]. VPN instances must be created in all BGP/MPLS IP VPN solutions. VPN instances support both the IPv4 and IPv6 address families.

VPN instance configuration consists of the following components :

o Per-Instance Configuration : that contains the common writable configuration objects for VPN instance IPv4 and IPv6 address family.

o Address Family Configuration of L3VPN Instance: that contains the address family specific writable configuration objects.

3.2.1. Per-Instance Configuration

This component contains the common writable configuration objects for VPN instance IPv4 and IPv6 address family.

```
              +--rw vpn-instances
              |  +--rw vpn-instance* [vpn-instance-name]
              |     +--rw vpn-instance-name    string
              |     +--rw description?         string
              |     +--rw ipv4-family
              ...
              |     +--rw ipv6-family
              ...
```

3.2.2. Address Family Configuration of L3VPN Instance

This component contains the address family specific writable configuration objects, such as route-distinguisher, vpnTargets, apply-label mode, etc.

```
          +--rw ipv4-family
          |  +--rw route-distinguisher?   string
          |  +--rw vpnTargets
          |  |  +--rw vpnTarget* [vrfRTValue]
          |  |     +--rw vrfRTValue    string
          |  |     +--rw vrfRTType     enumeration
          |  +--rw apply-label
          |  |  +--rw (apply-label-mode)?
          |  |     +--:(per-route)
          |  |     |  +--rw apply-label-per-route?      boolean
          |  |     +--:(per-instance)
          |  |        +--rw apply-label-per-instance?   boolean
          |  +--rw import-route-policy?   string
          |  +--rw export-route-policy?   string
          ...
          +--rw ipv6-family
             +--rw route-distinguisher?   string
             ...
```

3.3. VPN Interface Configuration

This component contains per-interface writable configuration objects, such as VPN instance binded, IPv4 address, IPv6 address, etc.

```
+--rw vpn-interfaces
|  +--rw vpn-interface* [name]
|     +--rw name                 leafref
|     +--rw vpn-instance-name?   string

...
```

3.4. MP-BGP Configuration for L3VPN

This component uses the definitions defined in BGP YANG module and augments some parameters. In a BGP/MPLS IP VPN, PEs must use MP-BGP to advertise VPNv4 or VPNv6 routes with the RD information to each other. A few of parameters have been defined in BGP YANG module [I-D.zhdankin-netmod-bgp-cfg]. This document adds some parameters.

```
augment /bgp:bgp-router/bgp:vpnv4/bgp:unicast:
   +--rw apply-label-per-nexthop?   boolean
   +--rw upeEnable?                 boolean
augment /bgp:bgp-router/bgp:vpnv6/bgp:unicast:
   +--rw apply-label-per-nexthop?   boolean
   +--rw upeEnable?                 boolean
...
```

3.5. BGP VPN Instance Configuration

In a BGP/MPLS IP VPN, a routing protocol or static routes must be configured between a PE and a CE to allow them to communicate and allow the CE to obtain routes to other CEs. The routing protocol can be EBGP, IBGP, RIP, OSPF, or IS-IS. Choose one of the following configurations as needed.

This section contains writable configuration objects when using BGP between PE and CE.

```
       augment /bgp:bgp-router:
          +--rw bgp-af-ipv4-vpn-instances
          |  +--rw bgp-af-ipv4-vpn-instance* [vpn-instance-name]
          |     +--rw vpn-instance-name    string
          |     +--rw router-id
          |     |  +--rw enable?               boolean
          |     |  +--rw (config-type)?
          |     |     +--:(static)
          |     |     |  +--rw ip-address?           inet:ip-address
          |     |     +--:(auto-select)
          |     |        +--rw enable-auto-select?   boolean
          |     +--rw auto-frr?            boolean
          |     +--rw bgpPeers
          |        +--rw bgpPeer* [peerAddr]
          |           +--rw peerAddr              inet:ip-address
          |           +--rw groupName?            string
          |           +--rw remoteAs?             string
          |           +--rw description?          string
          |           +--rw soo?                  string
          |           +--rw substituteAsEnable?   boolean
          +--rw bgp-af-ipv6-vpn-instances
             +--rw bgp-af-ipv6-vpn-instance* [vpn-instance-name]
                +--rw vpn-instance-name    string
                +--rw router-id
                |  +--rw enable?               boolean
                |  +--rw (config-type)?
                |     +--:(static)
                |     |  +--rw ip-address?           inet:ip-address
                |     +--:(auto-select)
                |        +--rw enable-auto-select?   boolean
                +--rw auto-frr?            boolean
                +--rw bgpPeers
                   +--rw bgpPeer* [peerAddr]
                      +--rw peerAddr              inet:ip-address
                      +--rw groupName?            string
                      +--rw remoteAs?             string
                      +--rw description?          string
                      +--rw soo?                  string
                      +--rw substituteAsEnable?   boolean
       ...
```

4. L3VPN YANG MODEL

L3VPN YANG MODEL

```
<CODE BEGINS> file "l3vpn@2014-08-15.yang"
module l3vpn {
  namespace "urn:huawei:params:xml:ns:yang:l3vpn";
```

```
  // replace with IANA namespace when assigned
  prefix "l3vpn";

  import bgp {
    prefix bgp;
    //draft-zhdankin-netmod-bgp-cfg
  }
  import ietf-interfaces {
    prefix if;
    //rfc7223-YANG Interface Management
  }

  import ietf-inet-types {
    prefix inet;
    //RFC6991
  }

  import ietf-yang-types {
    prefix yang;
    //RFC6991
  }

  description
    "This YANG module defines the generic configuration data for L3VPN service.

     Terms and Acronyms

     BGP (bgp): Border Gateway Protocol
     IPv4 (ipv4):Internet Protocol Version 4
     IPv6 (ipv6): Internet Protocol Version 6

    ";

  revision 2014-08-15 {
    description
      "Initial revision.";
      reference "RFC4271, RFC4364, RFC4760";
  }

  grouping augment-bgp-af-vpn-config {
    description
      "A set of configuration parameters that is applicable to both BGP-VPNv4
      and BGP-VPNv6 address family.";

    leaf apply-label-per-nexthop {
      description
        "The apply-label per-nexthop command enables the ASBR to allocate
         labels for IPv4 VPN routes or IPv6 VPN routes based on the next hop.";
```

```
      config "true";
      type boolean;
      default "false";
    }

    leaf upeEnable {
      description
        "Specify peer as UPE.";

      config "true";
      type boolean;
      default "false";
    }

  }


  grouping bgp-af-vpn-instance-config {

    container router-id {
      description
        "The router-id command configures router ID for BGP VPN instance IPv4
        or IPv6 address family.
        By default, no router ID is configured for BGP VPN instance IPv4 or
        IPv6 address family, and the BGP router ID is used as the router ID.";

      leaf enable {
        type boolean;
      }

      choice config-type {
        case static {
          leaf ip-address {
            description
              "Specifies the router ID of a BGP VPN instance IPv4 address
               family. The router ID is expressed in the IPv4 address format.
               ";

            config "true";
            type inet:ip-address;
          }
        }
        case auto-select {
          leaf enable-auto-select {
            description
              "Configures automatic route ID selection for the current BGP VPN
               instance address family.";
```

```
            config "true";
            type boolean;
          }
        }
      }
    }

    leaf auto-frr {
      description
        "The auto-frr command enables BGP Auto FRR.";

      config "true";
      type boolean;
      default "false";
    }

    container bgpPeers {
      list bgpPeer {
      key "peerAddr";
      max-elements "unbounded";
      min-elements "0";
        description
          "BGP Peer configure class";

        leaf peerAddr {
          description
            "The nerighbor address";
            config "true";
            type inet:ip-address;
            mandatory true;
        }

          leaf groupName {
            description "peerGroupName";
            config "true";
            type string {
              length "1..47";
            }
          }
          leaf remoteAs {
            description "Specifies the AS number of the peer.";
            config "true";
            type string {
              length "1..11";
            }
          }

          leaf description {
```

```
            description
              "specifies the description. The description is a string of letters
               or figures. The value ranges from 1 to 80 characters without
               spaces.";
          config "true";
            type string {
              length "1..80";
              pattern "([^?]*)";
            }
          }

          leaf soo {
            description
              "The peer soo command configures the Site of Origin (SoO)
               attribute for an EBGP peer in a BGP VPN instance. Format is ASN:n
n
               or IP-address:nn.";

          config "true";
          type string {
            length "3..21";
          }
        }

        leaf substituteAsEnable {
          description
            "Using the peer substitute-as command, you can substitute the AS
             number of the specified peer in the as-path with the local AS
             number.";

          config "true";
          type boolean;
            default "false";
        }

      }
    }

  }

  grouping vpn-af-config {
    description
      "A set of configuration parameters that is applicable to both IPv4 and
       IPv6 address family for a VPN instance .";

    leaf route-distinguisher {
      description
        "The route-distinguisher command configures a route distinguisher (RD)
         for the IPv4 or IPv6 address family of a VPN instance.
```

```
         Format is ASN:nn or IP-address:nn.";

      config "true";
      type string {
        length "3..21";
      }
    }

    container vpnTargets {
      description
        "The vpn-target command configures the export or import VPN target
         extended community attribute for the VPN instance IPv4/IPv6 address
         family.
         Format is ASN:nn or IP-address:nn.";

      list vpnTarget {
        key "vrfRTValue";
        max-elements "unbounded";
        min-elements "0";
        description
          "L3vpn vpntarget configure class";

        leaf vrfRTValue {

          description
            "Vpn-target: adds VPN target extended community attribute to the
             export or import VPN target extended community list. The
             vpn-target can be expressed in either of the following formats:
             (1)16-bit AS number:32-bit user-defined number
                 For example, 1:3. The AS number ranges from 0 to 65535. The
                 user-defined number ranges from 0 to 4294967295. The AS number
                 and the user-defined number cannot be 0s at the same time.
                 That is, a VPN target cannot be 0:0.
             (2)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. The user-defined number ranges from
                0 to 65535.
             (3)32-bit IP address:16-bit user-defined number
                For example, 192.168.122.15:1. An IP address ranges from
                0.0.0.0 to 255.255.255.255. A user-defined number ranges from 0
                to 65535.";

          config "true";
          mandatory "true";
          type string {
            length "3..21";
          }
        }
```

```
        leaf vrfRTType {
          description
            "Specifies the vpn target type, export-extcommunity:
             specifies the extended community attributes carried in routing
             information to be sent. import-extcommunity: receives routing
             information carrying specified extended community attributes.";

            mandatory "true";
            type enumeration {
            enum export_extcommunity {
              value "0";
              description "export-extcommunity:";
            }
            enum import_extcommunity {
              value "1";
              description "import-extcommunity:";
            }
            enum both {
              value "2";
              description "export-extcommunity & import-extcommunity:";
            }
          }
        }
      }

      container apply-label {
        description
          "Apply one label mode for the VPN instance route.";

        choice apply-label-mode {
          case per-route {
            description
              "The apply-label per-route command enables the one-label-per-route
               mode. The VPN instance IPv4/IPv6 address family assigns a unique
               label to each route to be sent to the peer PE.";

            leaf apply-label-per-route {
              type boolean;
              default "true";
            }
          }
          case per-instance {
            description
              "The apply-label per-instance command applies one label to all VPN
               instance IPv4 address family or IPv6 address family routes to a
               peer PE.";
```

```
          leaf apply-label-per-instance {
            type boolean;
            default "false";
          }
        }
      }
    }//End of "container apply-label"

    leaf import-route-policy {
      description
        "The import route-policy command associates a VPN instance enabled
         with the IPv4 or IPv6 address family with an import routing policy.
         Only one import routing policy can be associated with a VPN instance
         enabled with the IPv4 or IPv6 address family. If the import
         route-policy command is run more than once, the latest configuration
         overrides the previous ones.";

        config "true";
        type string {
          length "1..40";
        }
    }

    leaf export-route-policy {
      description
        "The export route-policy command associates a VPN instance enabled
         with the IPv4 or IPv6 address family with an export routing policy.
         Only one export routing policy can be associated with a VPN instance
         enabled with the IPv4 or IPv6 address family. If the export
         route-policy command is run more than once, the latest configuration
         overrides the previous ones.";

      config "true";
      type string {
        length "1..40";
      }
    }


    container prefix-limit {
      description
        "The prefix limit command sets a limit on the maximum number of
         prefixes supported in the existing VPN instance, preventing the
         PE from importing excessive VPN route prefixes.";

      leaf prefix-limit-number {
        description
          "Specifies the maximum number of prefixes supported in the VPN
```

```
           instance IPv4 or IPv6 address family.";

        type uint32 {
          range "1..4294967295";
        }
      }

      choice prefix-limit-action {
        case enable-alert-percent {
          leaf alert-percent-value {
            description
              "Specifies the proportion of the alarm threshold to the maximum
               number of prefixes.";
            type uint8 {
              range "1..100";
            }
          }
          leaf route-unchanged {
            description
              "Indicates that the routing table remains unchanged. By default,
               route-unchanged is not configured. When the number of prefixes
               in the routing table is greater than the value of the parameter
               number, routes are processed as follows:
               (1)If route-unchanged is configured, routes in the routing table
                  remain unchanged.
               (2)If route-unchanged is not configured, all routes in the
                  routing table are deleted and then re-added.";

            config "true";
            type boolean;
            default "false";
          }
        }
        case enable-simple-alert {
          leaf simple-alert {
            description
              "Indicates that when the number of VPN route prefixes exceeds
               number, prefixes can still join the VPN routing table and
               alarms are displayed.";

            config "true";
            type boolean;
            default "false";
          }
        }
      }
    }
```

```
    container routing-table-limit {
      description
        "The routing-table limit command sets a limit on the maximum number of
        routes that the IPv4 or IPv6 address family of a VPN instance can
        support.
        By default, there is no limit on the maximum number of routes that the
        IPv4 or IPv6 address family of a VPN instance can support, but the
        total number of private network and public network routes on a device
        cannot exceed the allowed maximum number of unicast routes.";

      leaf routing-table-limit-number {
        description
          "Specifies the maximum number of routes supported by a VPN instance.
          ";

        config "true";
        type uint32 {
          range "1..4294967295";
        }
      }
      choice routing-table-limit-action {
        case enable-alert-percent {
          leaf alert-percent-value {
            description
              "Specifies the percentage of the maximum number of routes. When
               the maximum number of routes that join the VPN instance is up
               to the value (number*alert-percent)/100, the system prompts
               alarms. The VPN routes can be still added to the routing table,
               but after the number of routes reaches number, the subsequent
               routes are dropped.";

            config "true";
            type uint8 {
              range "1..100";
            }
          }
        }
        case enable-simple-alert {
          leaf simple-alert {
            description
              "Indicates that when VPN routes exceed number, routes can still
               be added into the routing table, but the system prompts alarms.
               However, after the total number of VPN routes and network public
               routes reaches the unicast route limit specified in the License,
               the subsequent VPN routes are dropped.";

            config "true";
            type boolean;
```

```
          }
        }
      }
    }

    leaf vpn-frr {
      description
        "Enable VPN FRR in the VPN instance address family view.
         If a PE is connected to two other PEs, running the vpn frr command in
         the VPN instance address family view of the PE enables VPN FRR and
         improves network reliability. After VPN FRR is configured, traffic can
         switch to the secondary LSP immediately after the primary LSP becomes
         faulty.";

      type boolean;
      default "false";
    }


   /*
    * VPN QoS.
    */
    container l3vpnVrfPipe {
      description
        "The diffserv-mode command configures the mode of the MPLS
        differentiated service (Diff-Serv) for ensuring end-to-end QoS.";

      leaf pipeMode {
        description
          "Pipe mode";

    type enumeration {
      enum pipe {
        value "0";
        description
          "pipe: Indicates that the Pipe MPLS Diff-Serv mode is adopted.";
      }
      enum shortPipe {
        value "1";
        description
          "shortPipe: Indicates that the Short-pipe MPLS Diff-Serv mode
          is adopted.";
      }
      enum uniform {
        value "2";
        description
          "uniform: Indicates that the Uniform MPLS Diff-Serv mode is
          adopted.";
```

```
      }
    }
    default "uniform";

      }

      leaf serviceClass {
        description
          "Service Class, Specifies the service type when the packet enters the
          public network from the private network. The values are cs7, cs6, ef,
          af4, af3, af2, af1, be.";

          type enumeration {
            enum be {
              value "0";
              description "be:";
            }
            enum af1 {
              value "1";
              description "af1:";
            }
            enum af2 {
              value "2";
              description "af2:";
            }
            enum af3 {
              value "3";
              description "af3:";
            }
            enum af4 {
              value "4";
              description "af4:";
            }
            enum ef {
              value "5";
              description "ef:";
            }
            enum cs6 {
              value "6";
              description "cs6:";
            }
            enum cs7 {
              value "7";
              description "cs7:";
            }
          }
          default "be";
      }
```

```
      leaf color {
        description
          "Specifies a color for marking the discard priority of a packet
          transferred from a private network to a public network. The values
          are green, yellow, and red.";

        type enumeration {
          enum green {
            value "0";
            description "green:";
          }
          enum yellow {
            value "1";
            description "yellow:";
          }
          enum red {
            value "2";
            description "red:";
          }
        }
        default "green";
      }

      leaf dsName {
        description
          "Specifies the DS domain name of the specified Per-Hop Behavior (PHB)
          applied to the egress in Short pipe mode. It is a string of 1 to 31
          characters.";

        type string;
        default "default";
      }
    }

    container l3vpnTtlMode {
      description
        "The ttl-mode command enables MPLS to process the TTL in a specified
        mode. By default, MPLS processes the TTL in pipe mode.";

      leaf ttlMode {
        description "TTL mode";
        default "pipe";
        type enumeration {
          enum pipe {
            value "0";
            description
              "pipe: Enables MPLS to process the TTL in pipe mode.";
          }
```

```
          enum uniform {
            value "1";
            description
              "uniform: Enables MPLS to process the TTL in uniform mode.";
          }
        }
      }
    }

    leaf tunnel-policy {
      description
        "The tnl-policy command associates the IPv4 or IPv6 address family of
         a VPN instance with a tunnel policy.";

      type string {
        length "1..39";
      }
    }

    container importRibs {
      description
        "Import route class";

      leaf protocol {
        description
          "Specifies the protocol from which routes are imported.
          At present, In the IPv4 unicast address family view, the protocol
          can be IS-IS,static, direct and BGP.";

        type enumeration {
          enum ALL {
            value "0";
            description "ALL:";
          }
          enum Direct {
            value "1";
            description "Direct:";
          }
          enum OSPF {
            value "2";
            description "OSPF:";
          }
          enum ISIS {
            value "3";
            description "ISIS:";
          }
          enum Static {
            value "4";
```

```
            description "Static:";
          }
          enum RIP {
            value "5";
            description "RIP:";
          }
          enum BGP {
            value "6";
            description "BGP:";
          }
          enum OSPFV3 {
            value "7";
            description "OSPFV3:";
          }
          enum RIPNG {
            value "8";
            description "RIPNG:";
          }
          enum INVALID {
            value "9";
            description "INVALID:";
          }
        }
      }

      leaf processId {
        description
          "Specifies the process ID if the protocol from routes are imported is
           IS-IS.";

        default "0";
        type uint32 {
          range "0..4294967295";
        }
      }

      leaf bgp-valid-route {
        type boolean;
      }

      leaf policyName {
        description
          "Policy Id for import routes";
        type string {
        }
      }

    }
```

```
    leaf traffic-statistics {
      description
        "The traffic-statistics enable command enables traffic statistics
         for a VPN instance.";

      type boolean;
      default "false";
    }

  }


  /*
   * VPN instance view.
   */
  container vpn-instances {
    description
      "VPN instances configuration parameters.
       VPN instances support both the IPv4 and IPv6 address families.";

    list vpn-instance {
      max-elements "unbounded";
      min-elements "0";
      key "vpn-instance-name";
      description
        "Specifies the name of the VPN instance. It is a string of 1 to 31
         case-sensitive characters.";

      leaf vpn-instance-name {
        mandatory "true";
        type string {
          length "1..31";
        }
        description
          "The name of the vpn-instance.";
      }

      leaf description {
        description
          "A textual description of VPN instance, the VPN instance description
          helps users memorize the VPN instance.";

        type string {
          length "1..242";
          pattern "([^?]*)";
        }
      }
```

```
      container ipv4-family {
        description
          "The IPv4 address family is enabled for the VPN instance.";

        uses vpn-af-config;
      }

      container ipv6-family {
        description
          "The IPv6 address family is enabled for the VPN instance.";

        uses vpn-af-config;
      }


    }
  }



  /*
   * Binding Interfaces to a VPN Instance.
   */

  container vpn-interfaces {
    description
      "VPN is enabled on interfaces.";

    list vpn-interface  {
      key "name";
      max-elements "unbounded";
      min-elements "0";
      leaf name {
        type leafref {
          path "/if:interfaces/if:interface/if:name";
        }
      }
      leaf vpn-instance-name {
        type string {
          length "1..40";
        }
      }
    }
  }

  container vrfInfo {
    description
      "Display the information of the vrf.
```

```
      It is intended that this container may be augmented by vendors to
      reflect the vendor-specific operational state parameters.";

    leaf vrfCreateTime {
      description
        "CreateTime of the vrf.";
      config "false";
      type yang:timestamp;
    }

    leaf vrfUpTime {
      description
        "UpTime period of the vrf.";
      config "false";
      type yang:timeticks;
    }

    leaf label {
      description
        "Label of the vrf.";
      config "false";
      type uint32 {
        range "16..1048574";
      }
    }

    leaf vrfStatus {
      description
        "vrf status.";
      config "false";
      type enumeration {
        enum up {
          value "0";
          description "vrf up.";
        }
        enum down {
          value "1";
          description "vrf down.";
        }
      }
    }
  }


  /*
   * augment some bgp vpn functions in bgp module.
   */
  augment "/bgp:bgp-router/bgp:vpnv4/bgp:unicast" {
```

```
    uses augment-bgp-af-vpn-config;

  }

  augment "/bgp:bgp-router/bgp:vpnv6/bgp:unicast" {
    uses augment-bgp-af-vpn-config;

  }

  augment "/bgp:bgp-router" {

    container bgp-af-ipv4-vpn-instances {
      description
        "vpn-instances ipv4 address family.";
      list bgp-af-ipv4-vpn-instance {
        key "vpn-instance-name";
        max-elements "unbounded";
        min-elements "0";
        leaf vpn-instance-name {
          type string;
        }
        uses bgp-af-vpn-instance-config;
      }
    }

    container bgp-af-ipv6-vpn-instances {
      description
        "vpn-instances ipv6 address family.";
      list bgp-af-ipv6-vpn-instance {
        key "vpn-instance-name";
        max-elements "unbounded";
        min-elements "0";
        leaf vpn-instance-name {
          type string;
        }
        uses bgp-af-vpn-instance-config;
      }
    }

  }


}
</CODE ENDS>
```

5. IANA Considerations

This document makes no request of IANA.

6. Security Considerations

This document does not introduce any new security risk.

7. Acknowledgements

The authors would like to thank Guangying Zheng, Gang Yan for their contributions to this work.

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.

[RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.

[RFC4271] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[RFC4364] Rosen, E. and Y. Rekhter, "BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4364, February 2006.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.

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

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

[RFC6991] Schoenwaelder, J., "Common YANG Data Types", RFC 6991, July 2013.

8.2. Informative References

[I-D.zhdankin-netmod-bgp-cfg]
Alex, A., Patel, K., and A. Clemm, "Yang Data Model for BGP Protocol", draft-zhdankin-netmod-bgp-cfg-00 (work in progress), July 2013.

[RFC7223] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 7223, May 2014.

Authors' Addresses

Shunwan Zhuang
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: zhuangshunwan@huawei.com

Zhenbin Li
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: lizhenbin@huawei.com