

PIM Working Group
Internet-Draft
Intended Status: Standard Track
Expires: September 13, 2017

X. Liu
Jabil
F. Guo
Huawei
M. Sivakumar
Cisco
P. McAllister
Metaswitch Networks
A. Peter
Juniper Networks
March 13, 2017

A YANG data model for Internet Group Management Protocol (IGMP) and
Multicast Listener Discovery (MLD)
draft-ietf-pim-igmp-mld-yang-03

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-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 September 13, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This document defines a YANG data model that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices.

Table of Contents

1. Introduction	2
1.1. Requirements Language.....	3
1.2. Terminology	3
2. Design of Data model.....	3
2.1. Scope of model	3
2.2. Optional capabilities.....	3
2.3. Position of address family in hierarchy.....	4
3. Module Structure	4
3.1. IGMP and MLD Configuration.....	4
3.2. IGMP and MLD Operational State.....	8
3.3. IGMP and MLD RPC.....	12
4. IGMP and MLD YANG Modules.....	13
5. Security Considerations.....	38
6. IANA Considerations	38
7. References	39
7.1. Normative References.....	39
7.2. Informative References.....	40
8. Acknowledgments	40

1. Introduction

YANG [RFC6020] [RFC6087] is a data definition language that was introduced to model the configuration and running state of a device managed using NETCONF [RFC6241]. YANG is now also being used as a component of wider management interfaces, such as CLIs.

This document defines a draft YANG data model that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices. Currently this model is incomplete, but it will support the core IGMP and MLD protocols, as well as many other features mentioned in separate IGMP and MLD RFCs. Non-core features are defined as optional in the provided data model.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [RFC2119].

1.2. Terminology

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

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

2. Design of Data model

2.1. Scope of model

The model covers IGMPv1 [RFC1112], IGMPv2 [RFC2236], IGMPv3 [RFC3376] and MLDv1 [RFC2710], MLDv2 [RFC3810].

The representation of some of extension features is not specified in this draft of the data model. This model is being circulated in its current form for early oversight and review of the basic hierarchy.

The operational state fields and rpcs of this model can also be extended, though the structure of what has been written may be taken as representative of the structure of the whole model.

This model does not cover other IGMP and MLD related protocols such as IGMP/MLD Proxy [RFC4605] or IGMP/MLD Snooping [RFC4541] etc., these will be specified in separate documents.

2.2. Optional capabilities

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

There is also value in widely-supported features being standardized, to save work for individual vendors, and so that mapping between different vendors' configuration is not needlessly complicated. Therefore these modules declare a number of features representing capabilities that not all deployed devices support.

The extensive use of feature declarations should also substantially simplify the capability negotiation process for a vendor's IGMP and MLD implementations.

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

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

2.3. Position of address family in hierarchy

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

3. Module Structure

3.1. IGMP and MLD Configuration

The IGMP and MLD modules define the routing-control-plane-protocol-wide configuration options separately in a three-level hierarchy as listed below:

Global level: IGMP or MLD configuration attributes for the entire routing system

Interface-global: IGMP or MLD configuration attributes applicable to all interfaces IGMP MLD configuration attributes applied to interfaces whose interface level attributes are not existing, with same attributes' value for those

Interface-level: IGMP or MLD configuration attributes specific to the given interface

Where fields are not genuinely essential to protocol operation, they are marked as optional. Some fields will be essential but have a default specified, so that they need not be configured explicitly.

The module structure also applies, where applicable, to the operational state and notifications as well.

Our current direction is to agree to a protocol-centric model , and the IGMP and MLD model augments "/rt:routing/rt:control-plane-protocols/ rt:control-plane-protocol" and would allow a single protocol instance per VRF.

```

augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
  +--rw igmp
    +--rw global
      |   +--rw enable?          boolean {global-admin-enable}?
      |   +--rw max-entries?     uint32 {global-max-entries}?
      |   +--rw max-groups?      uint32 {global-max-groups}?
    +--rw interfaces
      +--rw (last-member-query-interval)?
        |   +--:(last-member-query-interval-basic)
        |   |   +--rw last-member-query-interval-basic?      uint16
        |   +--:(last-member-query-interval-extended)
        |   |   +--rw last-member-query-interval-extended?   uint16 {intf-las
t-member-query-interval-extended}?
        |   +--rw max-groups-per-interface?                  uint32 {intf-max-group
s}?
      +--rw (query-interval)?
        |   +--:(query-interval-basic)
        |   |   +--rw query-interval-basic?                  uint16
        |   +--:(query-interval-extended)
        |   |   +--rw query-interval-extended?               uint16 {intf-que
ry-interval-extended}?
        |   +--rw (query-max-response-time)?
        |   |   +--:(query-max-response-time-basic)
        |   |   |   +--rw query-max-response-time-basic?      uint16
        |   |   +--:(query-max-response-time-extended)
        |   |   |   +--rw query-max-response-time-extended?   uint16 {intf-que
ry-max-response-time-extended}?
        |   +--rw require-router-alert?                      boolean {intf-require-
router-alert}?
      +--rw (robustness-variable)?
        |   +--:(robustness-variable-basic)
        |   |   +--rw robustness-variable-basic?            uint8
        |   +--:(robustness-variable-extended)
        |   |   +--rw robustness-variable-extended?         uint8 {intf-robu
stness-variable-extended}?
        |   +--rw version?                                    uint8
      +--rw interface* [interface-name]
        +--rw interface-name                                if:interface-ref
        +--rw enable?                                       boolean {intf-admin
-enable}?
      +--rw group-policy?                                    string
      +--rw immediate-leave?                                empty {intf-immedia
te-leave}?
      +--rw (last-member-query-interval)?
        |   +--:(last-member-query-interval-basic)
        |   |   +--rw last-member-query-interval-basic?      uint16
        |   +--:(last-member-query-interval-extended)
        |   |   +--rw last-member-query-interval-extended?   uint16 {intf-
last-member-query-interval-extended}?
        |   +--rw max-groups?                                uint32 {intf-max-gr
oups}?

```

```

    oup-sources}?
        +--rw max-group-sources?                               uint32 {intf-max-gr
        +--rw (query-interval)?
            +--:(query-interval-basic)
            |   +--rw query-interval-basic?                     uint16
            |   +--:(query-interval-extended)
            |       +--rw query-interval-extended?              uint16 {intf-
query-interval-extended}?
            +--rw (query-max-response-time)?
            |   +--:(query-max-response-time-basic)
            |   |   +--rw query-max-response-time-basic?        uint16
            |   |   +--:(query-max-response-time-extended)
            |   |       +--rw query-max-response-time-extended?  uint16 {intf-
query-max-response-time-extended}?
            +--rw require-router-alert?                         boolean {intf-requi
re-router-alert}?
            +--rw (robustness-variable)?
            |   +--:(robustness-variable-basic)
            |   |   +--rw robustness-variable-basic?            uint8
            |   |   +--:(robustness-variable-extended)
            |   |       +--rw robustness-variable-extended?      uint8 {intf-r
obustness-variable-extended}?
            +--rw source-policy?                                string {intf-source
-policy}?
            +--rw verify-source-subnet?                         empty {intf-verify-
source-subnet}?
            +--rw version?                                       uint8
            +--rw join-group*                                     inet:ipv4-address {
intf-join-group}?
            +--rw ssm-map* [source-addr group-policy] {intf-ssm-map}?
            |   +--rw source-addr      ssm-map-ipv4-addr-type
            |   +--rw group-policy      string
            +--rw static-group* [group source-addr] {intf-static-group}?
            |   +--rw group              inet:ipv4-address
            |   +--rw source-addr        source-ipv4-addr-type
augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
    +--rw mld
        +--rw global
            +--rw enable?          boolean {global-admin-enable}?
            +--rw max-entries?      uint32 {global-max-entries}?
            +--rw max-groups?       uint32 {global-max-groups}?
        +--rw interfaces
            +--rw (last-member-query-interval)?
            |   +--:(last-member-query-interval-basic)
            |   |   +--rw last-member-query-interval-basic?      uint16
            |   |   +--:(last-member-query-interval-extended)
            |   |       +--rw last-member-query-interval-extended?  uint16 {intf-las
t-member-query-interval-extended}?
            +--rw max-groups-per-interface?                      uint32 {intf-max-group
s}?
            +--rw (query-interval)?
            |   +--:(query-interval-basic)
            |   |   +--rw query-interval-basic?                  uint16
            |   |   +--:(query-interval-extended)
            |   |       +--rw query-interval-extended?            uint16 {intf-que
ry-interval-extended}?
            +--rw (query-max-response-time)?
            |   +--:(query-max-response-time-basic)
            |   |   +--rw query-max-response-time-basic?          uint16
            |   |   +--:(query-max-response-time-extended)

```



```

|      +--rw query-max-response-time-extended?      uint16 {intf-que
ry-max-response-time-extended}?
|      +--rw require-router-alert?                  boolean {intf-require-
router-alert}?
|      +--rw (robustness-variable)?
|      |      +---:(robustness-variable-basic)
|      |      |      +--rw robustness-variable-basic?      uint8
|      |      |      +---:(robustness-variable-extended)
|      |      |      +--rw robustness-variable-extended?    uint8 {intf-robustness-variable-extended}?
|      |      +--rw version?                              uint8
|      +--rw interface* [interface-name]
|      |      +--rw interface-name                      if:interface-ref
|      |      +--rw enable?                              boolean {intf-admin
-enable}?
|      |      +--rw group-policy?                        string
|      |      +--rw immediate-leave?                     empty {intf-immediate-leave}?
|      |      +--rw (last-member-query-interval)?
|      |      |      +---:(last-member-query-interval-basic)
|      |      |      |      +--rw last-member-query-interval-basic?      uint16
|      |      |      |      +---:(last-member-query-interval-extended)
|      |      |      |      +--rw last-member-query-interval-extended?    uint16 {intf-last-member-query-interval-extended}?
|      |      |      +--rw max-groups?                          uint32 {intf-max-groups}?
|      |      +--rw max-group-sources?                      uint32 {intf-max-group-sources}?
|      |      +--rw (query-interval)?
|      |      |      +---:(query-interval-basic)
|      |      |      |      +--rw query-interval-basic?              uint16
|      |      |      |      +---:(query-interval-extended)
|      |      |      |      +--rw query-interval-extended?          uint16 {intf-query-interval-extended}?
|      |      +--rw (query-max-response-time)?
|      |      |      +---:(query-max-response-time-basic)
|      |      |      |      +--rw query-max-response-time-basic?      uint16
|      |      |      |      +---:(query-max-response-time-extended)
|      |      |      |      +--rw query-max-response-time-extended?    uint16 {intf-query-max-response-time-extended}?
|      |      +--rw require-router-alert?                  boolean {intf-require-router-alert}?
|      |      +--rw (robustness-variable)?
|      |      |      +---:(robustness-variable-basic)
|      |      |      |      +--rw robustness-variable-basic?          uint8
|      |      |      |      +---:(robustness-variable-extended)
|      |      |      |      +--rw robustness-variable-extended?      uint8 {intf-robustness-variable-extended}?
|      |      +--rw source-policy?                          string {intf-source-policy}?
|      |      +--rw verify-source-subnet?                    empty {intf-verify-source-subnet}?
|      |      +--rw version?                                uint8
|      +--rw join-group*
|      |      +--rw ssm-map* [source-addr group-policy] {intf-ssm-map}?
|      |      |      +--rw source-addr      ssm-map-ipv6-addr-type
|      |      |      +--rw group-policy     string
|      |      +--rw static-group* [group source-addr] {intf-static-group}?
|      |      |      +--rw group            inet:ipv6-address
|      |      |      +--rw source-addr      source-ipv6-addr-type

```


3.2. IGMP and MLD Operational State

The IGMP or MLD module contains operational state information also in a three-level hierarchy as mentioned earlier and separately listed as below.

Global level: IGMP or MLD operational state attributes for the entire routing system

Interface-global: IGMP or MLD interface level operational state attributes applied to interfaces whose interface level attributes do not exist, with same attributes' value for those interfaces

Interface-specific: IGMP or MLD operational state attributes specific to the given interface.

```
augment /rt:routing-state/rt:control-plane-protocols/rt:control-plane-protocol:
  col:
    +--ro igmp
      +--ro global
        +--ro enable?          boolean {global-admin-enable}?
        +--ro max-entries?     uint32 {global-max-entries}?
        +--ro max-groups?      uint32 {global-max-groups}?
        +--ro entries-count?   uint32
        +--ro groups-count?    uint32
        +--ro statistics
          +--ro discontinuity-time?  yang:date-and-time
          +--ro error
            +--ro total?          yang:counter64
            +--ro query?          yang:counter64
            +--ro report?         yang:counter64
            +--ro leave?          yang:counter64
            +--ro checksum?       yang:counter64
            +--ro too-short?      yang:counter64
          +--ro received
            +--ro total?          yang:counter64
            +--ro query?          yang:counter64
            +--ro report?         yang:counter64
            +--ro leave?          yang:counter64
          +--ro sent
            +--ro total?          yang:counter64
            +--ro query?          yang:counter64
            +--ro report?         yang:counter64
            +--ro leave?          yang:counter64
      +--ro interfaces
        +--ro (last-member-query-interval)?
          +--:(last-member-query-interval-basic)
            +--ro last-member-query-interval-basic?      uint16
          +--:(last-member-query-interval-extended)
            +--ro last-member-query-interval-extended?  uint16 {intf-last-member-query-interval-extended}?
```

```

s}?
    +--ro max-groups-per-interface?                uint32 {intf-max-group
ry-interval-extended}?
    +--ro (query-interval)?
    |   +--:(query-interval-basic)
    |   |   +--ro query-interval-basic?                uint16
    |   +--:(query-interval-extended)
    |   |   +--ro query-interval-extended?            uint16 {intf-que
ry-max-response-time-extended}?
    +--ro (query-max-response-time)?
    |   +--:(query-max-response-time-basic)
    |   |   +--ro query-max-response-time-basic?        uint16
    |   +--:(query-max-response-time-extended)
    |   |   +--ro query-max-response-time-extended?    uint16 {intf-que
router-alert}?
    +--ro require-router-alert?                    boolean {intf-require-
robustness-variable-extended}?
    +--ro (robustness-variable)?
    |   +--:(robustness-variable-basic)
    |   |   +--ro robustness-variable-basic?            uint8
    |   +--:(robustness-variable-extended)
    |   |   +--ro robustness-variable-extended?        uint8 {intf-robustness-variable-extended}?
    +--ro version?                                uint8
    +--ro interface* [interface-name]
    |   +--ro interface-name                        if:interface-ref
    |   +--ro enable?                              boolean {intf-admin
te-leave}?
    +--ro group-policy?                            string
    +--ro immediate-leave?                         empty {intf-immedia
last-member-query-interval-extended}?
    +--ro (last-member-query-interval)?
    |   +--:(last-member-query-interval-basic)
    |   |   +--ro last-member-query-interval-basic?    uint16
    |   +--:(last-member-query-interval-extended)
    |   |   +--ro last-member-query-interval-extended? uint16 {intf-
max-groups?
    +--ro max-groups?                              uint32 {intf-max-gr
oup-sources}?
    +--ro max-group-sources?                       uint32 {intf-max-gr
query-interval-extended}?
    +--ro (query-interval)?
    |   +--:(query-interval-basic)
    |   |   +--ro query-interval-basic?                uint16
    |   +--:(query-interval-extended)
    |   |   +--ro query-interval-extended?            uint16 {intf-
query-max-response-time-extended}?
    +--ro (query-max-response-time)?
    |   +--:(query-max-response-time-basic)
    |   |   +--ro query-max-response-time-basic?        uint16
    |   +--:(query-max-response-time-extended)
    |   |   +--ro query-max-response-time-extended?    uint16 {intf-
require-router-alert}?
    +--ro require-router-alert?                    boolean {intf-requi
robustness-variable-extended}?
    +--ro (robustness-variable)?
    |   +--:(robustness-variable-basic)
    |   |   +--ro robustness-variable-basic?            uint8
    |   +--:(robustness-variable-extended)
    |   |   +--ro robustness-variable-extended?        uint8 {intf-r
-source-policy?
    +--ro source-policy?                           string {intf-source
verify-source-subnet?
    +--ro verify-source-subnet?                     empty {intf-verify-

```

source-subnet}?

+--ro version?

uint8

Liu & Guo, etc

Expires September 13, 2017

[Page 9]

```

+--ro join-group*                               inet:ipv4-address {
intf-join-group}?
+--ro ssm-map* [source-addr group-policy] {intf-ssm-map}?
|   +--ro source-addr          ssm-map-ipv4-addr-type
|   +--ro group-policy         string
+--ro static-group* [group source-addr] {intf-static-group}?
|   +--ro group                inet:ipv4-address
|   +--ro source-addr          source-ipv4-addr-type
+--ro oper-status?                enumeration
+--ro querier?                    inet:ipv4-address
+--ro joined-group*               inet:ipv4-address {
intf-join-group}?
+--ro group* [address]
|   +--ro address              inet:ipv4-address
|   +--ro expire?              uint32
|   +--ro filter-mode?         enumeration
|   +--ro host-count?          uint32
|   +--ro up-time?             uint32
|   +--ro host*                inet:ipv4-address
|   +--ro last-reporter?       inet:ipv4-address
|   +--ro source* [address]
|       +--ro address          inet:ipv4-address
|       +--ro expire?          uint32
|       +--ro up-time?         uint32
|       +--ro last-reporter?   inet:ipv4-address
augment /rt:routing-state/rt:control-plane-protocols/rt:control-plane-protocols:
col:
+--ro mld
|   +--ro global
|       +--ro enable?          boolean {global-admin-enable}?
|       +--ro max-entries?     uint32 {global-max-entries}?
|       +--ro max-groups?      uint32 {global-max-groups}?
|       +--ro entries-count?    uint32
|       +--ro groups-count?     uint32
|       +--ro statistics
|           +--ro discontinuity-time? yang:date-and-time
|           +--ro error
|               +--ro total?      yang:counter64
|               +--ro query?      yang:counter64
|               +--ro report?     yang:counter64
|               +--ro leave?      yang:counter64
|               +--ro checksum?   yang:counter64
|               +--ro too-short?  yang:counter64
|       +--ro received
|           +--ro total?          yang:counter64
|           +--ro query?          yang:counter64
|           +--ro report?         yang:counter64
|           +--ro leave?          yang:counter64
|       +--ro sent
|           +--ro total?          yang:counter64
|           +--ro query?          yang:counter64
|           +--ro report?         yang:counter64

```

```

|          +--ro leave?      yang:counter64
+--ro interfaces
  +--ro (last-member-query-interval)?
    |   +--:(last-member-query-interval-basic)
    |   |   +--ro last-member-query-interval-basic?      uint16
    |   |   +--:(last-member-query-interval-extended)
    |   |   +--ro last-member-query-interval-extended?    uint16 {intf-las
t-member-query-interval-extended}?
    +--ro max-groups-per-interface?      uint32 {intf-max-group
s}?
    +--ro (query-interval)?
      |   +--:(query-interval-basic)
      |   |   +--ro query-interval-basic?      uint16
      |   |   +--:(query-interval-extended)
      |   |   +--ro query-interval-extended?    uint16 {intf-que
ry-interval-extended}?
      +--ro (query-max-response-time)?
        |   +--:(query-max-response-time-basic)
        |   |   +--ro query-max-response-time-basic?      uint16
        |   |   +--:(query-max-response-time-extended)
        |   |   +--ro query-max-response-time-extended?    uint16 {intf-que
ry-max-response-time-extended}?
        +--ro require-router-alert?      boolean {intf-require-
router-alert}?
        +--ro (robustness-variable)?
          |   +--:(robustness-variable-basic)
          |   |   +--ro robustness-variable-basic?      uint8
          |   |   +--:(robustness-variable-extended)
          |   |   +--ro robustness-variable-extended?    uint8 {intf-robustness-variable-extended}?
          +--ro version?      uint8
        +--ro interface* [interface-name]
          +--ro interface-name      if:interface-ref
          +--ro enable?      boolean {intf-admin
-enable}?
          +--ro group-policy?      string
          +--ro immediate-leave?    empty {intf-immedia
te-leave}?
          +--ro (last-member-query-interval)?
            |   +--:(last-member-query-interval-basic)
            |   |   +--ro last-member-query-interval-basic?      uint16
            |   |   +--:(last-member-query-interval-extended)
            |   |   +--ro last-member-query-interval-extended?    uint16 {intf-
last-member-query-interval-extended}?
            +--ro max-groups?      uint32 {intf-max-gr
oups}?
            +--ro max-group-sources?      uint32 {intf-max-gr
oup-sources}?
            +--ro (query-interval)?
              |   +--:(query-interval-basic)
              |   |   +--ro query-interval-basic?      uint16
              |   |   +--:(query-interval-extended)
              |   |   +--ro query-interval-extended?    uint16 {intf-
query-interval-extended}?
              +--ro (query-max-response-time)?
                |   +--:(query-max-response-time-basic)
                |   |   +--ro query-max-response-time-basic?      uint16
                |   |   +--:(query-max-response-time-extended)
                |   |   +--ro query-max-response-time-extended?    uint16 {intf-
query-max-response-time-extended}?
                +--ro require-router-alert?      boolean {intf-requi
re-router-alert}?
                +--ro (robustness-variable)?

```



```

|   +---:(robustness-variable-basic)
|   |   +---ro robustness-variable-basic?           uint8
|   +---:(robustness-variable-extended)
|       +---ro robustness-variable-extended?       uint8 {intf-r
robustness-variable-extended}?
+---ro source-policy?                               string {intf-source
-policy}?
+---ro verify-source-subnet?                         empty {intf-verify-
source-subnet}?
+---ro version?                                     uint8
+---ro join-group*                                  inet:ipv6-address {
intf-join-group}?
+---ro ssm-map* [source-addr group-policy] {intf-ssm-map}?
|   +---ro source-addr      ssm-map-ipv6-addr-type
|   +---ro group-policy     string
+---ro static-group* [group source-addr] {intf-static-group}?
|   +---ro group            inet:ipv6-address
|   +---ro source-addr      source-ipv6-addr-type
+---ro oper-status?                               enumeration
+---ro querier?                                   inet:ipv6-address
+---ro joined-group*                              inet:ipv6-address {
intf-join-group}?
+---ro group* [address]
|   +---ro address          inet:ipv6-address
|   +---ro expire?          uint32
|   +---ro filter-mode?     enumeration
|   +---ro host-count?      uint32
|   +---ro up-time?         uint32
|   +---ro host*            inet:ipv6-address
|   +---ro last-reporter?   inet:ipv6-address
+---ro source* [address]
|   +---ro address          inet:ipv6-address
|   +---ro expire?          uint32
|   +---ro up-time?         uint32
|   +---ro last-reporter?   inet:ipv6-address

```

3.3. IGMP and MLD RPC

```

rpcs:

+---x clear-igmp-groups {rpc-clear-groups}?

|   +---w input

|       +---w interface?  -> /rt:routing/control-plane-protocols/control-p
lane-protocol/igmp/interfaces/interface/interface-name

|       +---w group?      inet:ipv4-address

|       +---w source?     inet:ipv4-address

+---x clear-mlld-groups {rpc-clear-groups}?

```



```
+---w input

+---w interface?  -> /rt:routing/control-plane-protocols/control-p
lane-protocol/mlld/interfaces/interface/interface-name

+---w group?      inet:ipv6-address

+---w source?     inet:ipv6-address
```

4. IGMP and MLD YANG Modules

```
<CODE BEGINS> file "ietf-igmp-mld@2017-03-13.yang"
module ietf-igmp-mld {
  namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld";
  // replace with IANA namespace when assigned
  prefix igmp-mld;

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

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

  import ietf-routing {
    prefix "rt";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-ip {
    prefix ip;
  }

  organization
    "IETF PIM Working Group";

  contact
    "WG Web:    <http://tools.ietf.org/wg/pim/>
    WG List:    <mailto:pim@ietf.org>

    WG Chair: Stig Venaas
               <mailto:stig@venaas.com>
```

WG Chair: Mike McBride
<mailto:mmcbride7@gmail.com>

Editor: Xufeng Liu
<mailto:Xufeng_Liu@jabil.com>

Editor: Feng Guo
<mailto:guofeng@huawei.com>

Editor: Mahesh Sivakumar
<mailto:masivaku@cisco.com>

Editor: Pete McAllister
<mailto:pete.mcallister@metaswitch.com>

Editor: Anish Peter
<mailto:anish.ietf@gmail.com>;

```
description
  "The module defines a collection of YANG definitions common for
  IGMP and MLD.";

revision 2017-03-13 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for IGMP and MLD";
}

/*
 * Features
 */
feature global-admin-enable {
  description
    "Support global configuration to enable or disable protocol.";
}

feature global-interface-config {
  description
    "Support global configuration applied for all interfaces.";
}

feature global-max-entries {
  description
    "Support configuration of global max-entries.";
}

feature global-max-groups {
```

```
    description
      "Support configuration of global max-groups.";
  }

  feature intf-admin-enable {
    description
      "Support configuration of interface administrative enabling.";
  }

  feature intf-last-member-query-interval-extended {
    description
      "Support configuration of interface last member query interval
      different value range.";
  }

  feature intf-query-interval-extended {
    description
      "Support configuration of interface query interval
      different value range.";
  }

  feature intf-query-max-response-time-extended {
    description
      "Support configuration of interface query max response time
      different value range.";
  }

  feature intf-robustness-variable-extended {
    description
      "Support configuration of interface robustness-variable
      different value range.";
  }

  feature intf-immediate-leave {
    description
      "Support configuration of interface immediate-leave.";
  }

  feature intf-join-group {
    description
      "Support configuration of interface join-group.";
  }

  feature intf-max-groups {
    description
      "Support configuration of interface max-groups.";
  }

  feature intf-max-group-sources {
```

```
    description
      "Support configuration of interface max-group-sources.";
  }

  feature intf-require-router-alert {
    description
      "Support configuration of interface require-router-alert.";
  }

  feature intf-source-policy {
    description
      "Support configuration of interface source policy.";
  }

  feature intf-ssm-map {
    description
      "Support configuration of interface ssm-map.";
  }

  feature intf-static-group {
    description
      "Support configuration of interface static-group.";
  }

  feature intf-verify-source-subnet {
    description
      "Support configuration of interface verify-source-subnet.";
  }

  feature per-interface-config {
    description
      "Support per interface configuration.";
  }

  feature rpc-clear-groups {
    description
      "Support rpc's to clear groups.";
  }

/*
 * Typedefs
 */
typedef ssm-map-ipv4-addr-type {
  type union {
    type enumeration {
      enum 'policy' {
        description
          "Source address is specified in SSM map policy.";
      }
    }
  }
}
```

```
    }
    type inet:ipv4-address;
  }
  description
    "Multicast source IP address type for SSM map.";
} // source-ipv4-addr-type

typedef ssm-map-ipv6-addr-type {
  type union {
    type enumeration {
      enum 'policy' {
        description
          "Source address is specified in SSM map policy.";
      }
    }
    type inet:ipv6-address;
  }
  description
    "Multicast source IP address type for SSM map.";
} // source-ipv6-addr-type

typedef source-ipv4-addr-type {
  type union {
    type enumeration {
      enum '*' {
        description
          "Any source address.";
      }
    }
    type inet:ipv4-address;
  }
  description
    "Multicast source IP address type.";
} // source-ipv4-addr-type

typedef source-ipv6-addr-type {
  type union {
    type enumeration {
      enum '*' {
        description
          "Any source address.";
      }
    }
    type inet:ipv6-address;
  }
  description
    "Multicast source IP address type.";
} // source-ipv6-addr-type
```

```
/*
 * Identities
 */

/*
 * Groupings
 */
grouping global-config-attributes {
  description "Global IGMP and MLD configuration.";

  leaf enable {
    if-feature global-admin-enable;
    type boolean;
    description
      "true to enable IGMP in the routing instance;
       false to disable IGMP in the routing instance.";
  }

  leaf max-entries {
    if-feature global-max-entries;
    type uint32;
    description
      "The maximum number of entries in IGMP.";
  }

  leaf max-groups {
    if-feature global-max-groups;
    type uint32;
    description
      "The maximum number of groups that IGMP can join.";
  }
} // global-config-attributes

grouping global-state-attributes {
  description "Global IGMP and MLD state attributes.";

  leaf entries-count {
    type uint32;
    description
      "The number of entries in IGMP.";
  }

  leaf groups-count {
    type uint32;
    description
      "The number of groups that IGMP can join.";
  }

  container statistics {
    description "Global statistics.";
  }
}
```

```
    leaf discontinuity-time {
        type yang:date-and-time;
        description
            "The time on the most recent occasion at which any one
            or more of the statistic counters suffered a
            discontinuity. If no such discontinuities have occurred
            since the last re-initialization of the local
            management subsystem, then this node contains the time
            the local management subsystem re-initialized itself.";
    }

    container error {
        description "Statistics of errors.";
        uses global-statistics-error;
    }

    container received {
        description "Statistics of received messages.";
        uses global-statistics-sent-received;
    }
    container sent {
        description "Statistics of sent messages.";
        uses global-statistics-sent-received;
    }
} // statistics
} // global-state-attributes

grouping global-statistics-error {
    description
        "A grouping defining statistics attributes for errors.";
    uses global-statistics-sent-received;
    leaf checksum {
        type yang:counter64;
        description
            "The number of checksum errors.";
    }
    leaf too-short {
        type yang:counter64;
        description
            "The number of messages that are too short.";
    }
} // global-statistics-error

grouping global-statistics-sent-received {
    description
        "A grouping defining statistics attributes.";
    leaf total {
        type yang:counter64;
        description
```

```
        "The number of total messages.";
    }
    leaf query {
        type yang:counter64;
        description
            "The number of query messages.";
    }
    leaf report {
        type yang:counter64;
        description
            "The number of report messages.";
    }
    leaf leave {
        type yang:counter64;
        description
            "The number of leave messages.";
    }
} // global-statistics-sent-received

grouping interfaces-config-attributes {
    description
        "Configuration attributes applied to interfaces whose
        per interface attributes are not existing.";

    choice last-member-query-interval {
        description
            "Different vendors can restrict different range to the
            Last Member Query Interval parameter.";

        leaf last-member-query-interval-basic {
            type uint16 {
                range "1..65535";
            }
            units seconds;
            default 1;
            description
                "Last Member Query Interval, which may be tuned to modify the
                leave latency of the network.";
            reference "RFC3376. Sec. 8.8.";
        }
        leaf last-member-query-interval-extended {
            if-feature intf-last-member-query-interval-extended;
            type uint16;
            units seconds;
            default 1;
            description
                "Last Member Query Interval, which may be tuned to modify the
                leave latency of the network.";
            reference "RFC3376. Sec. 8.8.";
        }
    }
}
```



```
    }  
  }  
  leaf max-groups-per-interface {  
    if-feature intf-max-groups;  
    type uint32;  
    description  
      "The maximum number of groups that IGMP can join.";  
  }  
  choice query-interval {  
    description  
      "Different vendors can restrict different range to the  
      Query Interval parameter.";  
  
    leaf query-interval-basic {  
      type uint16 {  
        range "1..31744";  
      }  
      units seconds;  
      default 125;  
      description  
        "The Query Interval is the interval between General Queries  
        sent by the Querier.";  
      reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";  
    }  
    leaf query-interval-extended {  
      if-feature intf-query-interval-extended;  
      type uint16;  
      units seconds;  
      default 125;  
      description  
        "The Query Interval is the interval between General Queries  
        sent by the Querier.";  
      reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";  
    }  
  }  
  choice query-max-response-time {  
    description  
      "Different vendors can restrict different range to the  
      Query maximum response time parameter.";  
  
    leaf query-max-response-time-basic {  
      type uint16 {  
        range "1..65535";  
      }  
      units seconds;  
      default 10;  
      description  
        "Query maximum response time specifies the maximum time  
        allowed before sending a responding report.";
```

```
        reference "RFC3376. Sec. 4.1.1, 8.3, 8.14.3.";
    }
    leaf query-max-response-time-extended {
        if-feature intf-query-max-response-time-extended;
        type uint16;
        units seconds;
        default 10;
        description
            "Query maximum response time specifies the maximum time
             allowed before sending a responding report.";
        reference "RFC3376. Sec. 4.1.1, 8.3, 8.14.3.";
    }
}
leaf require-router-alert {
    if-feature intf-require-router-alert;
    type boolean;
    default false;
    description
        "Protocol packets should contain router alert IP option.";
}
choice robustness-variable {
    description
        "Different vendors can restrict different range to the
         Robustness Variable parameter.";

    leaf robustness-variable-basic {
        type uint8 {
            range "2..7";
        }
        default 2;
        description
            "Querier's Robustness Variable allows tuning for the expected
             packet loss on a network.";
        reference "RFC3376. Sec. 4.1.6, 8.1, 8.14.1.";
    }
    leaf robustness-variable-extended {
        if-feature intf-robustness-variable-extended;
        type uint8;
        default 2;
        description
            "Querier's Robustness Variable allows tuning for the expected
             packet loss on a network.";
        reference "RFC3376. Sec. 4.1.6, 8.1, 8.14.1.";
    }
}
leaf version {
    type uint8 {
        range "1..3";
    }
}
```

```
        description "IGMP version.";
        reference "RFC1112, RFC2236, RFC3376.";
    }
} // interfaces-config-attributes

grouping interface-config-attributes-igmp {
    description "Per interface igmp configuration for IGMP.";

    uses interface-config-attributes-igmp-mld;

    leaf-list join-group {
        if-feature intf-join-group;
        type inet:ipv4-address;
        description
            "The router joins this multicast group on the interface.";
    }

    list ssm-map {
        if-feature intf-ssm-map;
        key "source-addr group-policy";
        description "The policy for (*,G) mapping to (S,G).";
        leaf source-addr {
            type ssm-map-ipv4-addr-type;
            description
                "Multicast source IP address.";
        }
        leaf group-policy {
            type string;
            description
                "Name of the access policy used to filter IGMP
                membership. A device MAY restrict the length
                and value of this name, possibly space and special
                characters are not allowed.";
        }
    }
}

list static-group {
    if-feature intf-static-group;
    key "group source-addr";
    description
        "A static multicast route, (*,G) or (S,G).";

    leaf group {
        type inet:ipv4-address;
        description
            "Multicast group IP address.";
    }
    leaf source-addr {
        type source-ipv4-addr-type;
    }
}
```

```
        description
            "Multicast source IP address.";
    }
}
} // interface-config-attributes-igmp

grouping interface-config-attributes-igmp-mld {
    description
        "Per interface configuration for both IGMP and MLD.";

    leaf enable {
        if-feature intf-admin-enable;
        type boolean;
        default false;
        description
            "true to enable IGMP on the interface;
             false to disable IGMP on the interface.";
    }
    leaf group-policy {
        type string;
        description
            "Name of the access policy used to filter IGMP
             membership.A device MAY restrict the length
             and value of this name, possibly space and special
             characters are not allowed.";
    }
    leaf immediate-leave {
        if-feature intf-immediate-leave;
        type empty;
        description
            "If present, IGMP perform an immediate leave upon receiving an
             IGMP Version 2 (IGMPv2) leave message.
             If the router is IGMP-enabled, it sends an IGMP last member
             query with a last member query response time. However, the
             router does not wait for the response time before it prunes
             off the group.";
    }
    choice last-member-query-interval {
        description
            "Different vendors can restrict different range to the
             Last Member Query Interval parameter.";

        leaf last-member-query-interval-basic {
            type uint16 {
                range "1..65535";
            }
            units seconds;
            default 1;
            description
```

```
        "Last Member Query Interval, which may be tuned to modify the
        leave latency of the network.";
        reference "RFC3376. Sec. 8.8.";
    }
    leaf last-member-query-interval-extended {
        if-feature intf-last-member-query-interval-extended;
        type uint16;
        units seconds;
        default 1;
        description
            "Last Member Query Interval, which may be tuned to modify the
            leave latency of the network.";
        reference "RFC3376. Sec. 8.8.";
    }
}
leaf max-groups {
    if-feature intf-max-groups;
    type uint32;
    description
        "The maximum number of groups that IGMP can join.";
}
leaf max-group-sources {
    if-feature intf-max-group-sources;
    type uint32;
    description
        "The maximum number of group sources.";
}
choice query-interval {
    description
        "Different vendors can restrict different range to the
        Query Interval parameter.";

    leaf query-interval-basic {
        type uint16 {
            range "1..31744";
        }
        units seconds;
        default 125;
        description
            "The Query Interval is the interval between General Queries
            sent by the Querier.";
        reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";
    }
    leaf query-interval-extended {
        if-feature intf-query-interval-extended;
        type uint16;
        units seconds;
        default 125;
        description
```

```
        "The Query Interval is the interval between General Queries
        sent by the Querier.";
        reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";
    }
}
choice query-max-response-time {
    description
        "Different vendors can restrict different range to the
        Query maximum response time parameter.";

    leaf query-max-response-time-basic {
        type uint16 {
            range "1..65535";
        }
        units seconds;
        default 10;
        description
            "Query maximum response time specifies the maximum time
            allowed before sending a responding report.";
        reference "RFC3376. Sec. 4.1.1, 8.3, 8.14.3.";
    }
    leaf query-max-response-time-extended {
        if-feature intf-query-max-response-time-extended;
        type uint16;
        units seconds;
        default 10;
        description
            "Query maximum response time specifies the maximum time
            allowed before sending a responding report.";
        reference "RFC3376. Sec. 4.1.1, 8.3, 8.14.3.";
    }
}
leaf require-router-alert {
    if-feature intf-require-router-alert;
    type boolean;
    description
        "Protocol packets should contain router alert IP option.";
}
choice robustness-variable {
    description
        "Different vendors can restrict different range to the
        Robustness Variable parameter.";

    leaf robustness-variable-basic {
        type uint8 {
            range "2..7";
        }
        default 2;
        description
```

```
        "Querier's Robustness Variable allows tuning for the expected
        packet loss on a network.";
        reference "RFC3376. Sec. 4.1.6, 8.1, 8.14.1.";
    }
    leaf robustness-variable-extended {
        if-feature intf-robustness-variable-extended;
        type uint8;
        default 2;
        description
            "Querier's Robustness Variable allows tuning for the expected
            packet loss on a network.";
        reference "RFC3376. Sec. 4.1.6, 8.1, 8.14.1.";
    }
}
leaf source-policy {
    if-feature intf-source-policy;
    type string;
    description
        "Name of the access policy used to filter sources.
        A device MAY restrict the length
        and value of this name, possibly space and special
        characters are not allowed.";
}
leaf verify-source-subnet {
    if-feature intf-verify-source-subnet;
    type empty;
    description
        "If present, the interface accepts packets with matching
        source IP subnet only.";
}
leaf version {
    type uint8 {
        range "1..3";
    }
    description "IGMP version.";
    reference "RFC1112, RFC2236, RFC3376.";
}
} // interface-config-attributes-igmp-mld

grouping interface-config-attributes-mld {
    description "Per interface mld configuration for MLD.";

    uses interface-config-attributes-igmp-mld;

    leaf-list join-group {
        if-feature intf-join-group;
        type inet:ipv6-address;
        description
            "The router joins this multicast group on the interface.";
    }
}
```

```
    }

    list ssm-map {
      if-feature intf-ssm-map;
      key "source-addr group-policy";
      description "The policy for (*,G) mapping to (S,G).";
      leaf source-addr {
        type ssm-map-ipv6-addr-type;
        description
          "Multicast source IPv6 address.";
      }
      leaf group-policy {
        type string;
        description
          "Name of the access policy used to filter MLD
           membership. A device MAY restrict the length
           and value of this name, possibly space and special
           characters are not allowed.";
      }
    }
  }

  list static-group {
    if-feature intf-static-group;
    key "group source-addr";
    description
      "A static multicast route, (*,G) or (S,G).";

    leaf group {
      type inet:ipv6-address;
      description
        "Multicast group IPv6 address.";
    }
    leaf source-addr {
      type source-ipv6-addr-type;
      description
        "Multicast source IPv6 address.";
    }
  }
} // interface-config-attributes-mlld

grouping interface-state-attributes-igmp {
  description
    "Per interface state attributes for IGMP.";

  uses interface-state-attributes-igmp-mlld;

  leaf querier {
    type inet:ipv4-address;
    description "The querier address in the subnet";
  }
}
```



```
}
leaf-list joined-group {
  if-feature intf-join-group;
  type inet:ipv4-address;
  description
    "The routers that joined this multicast group.";
}

list group {
  key "address";
  description
    "Multicast group membership information
    that joined on the interface.";

  leaf address {
    type inet:ipv4-address;
    description
      "Multicast group address.";
  }
  uses interface-state-group-attributes-igmp-mld;
  leaf-list host {
    type inet:ipv4-address;
    description
      "List of host address that
      joined the multicast group";
  }
  leaf last-reporter {
    type inet:ipv4-address;
    description
      "The last host address which has sent the
      report to join the multicast group.";
  }
  list source {
    key "address";
    description
      "List of multicast source information
      of the multicast group.";

    leaf address {
      type inet:ipv4-address;
      description
        "Multicast source address";
    }
  }
  uses interface-state-source-attributes-igmp-mld;
  leaf last-reporter {
    type inet:ipv4-address;
    description
      "The last host address which has sent the
      report to join the multicast source and group.";
```

```
    }
  } // list source
} // list group
} // interface-state-attributes-igmp

grouping interface-state-attributes-igmp-mld {
  description
    "Per interface state attributes for both IGMP and MLD.";

  leaf oper-status {
    type enumeration {
      enum up {
        description
          "Ready to pass packets.";
      }
      enum down {
        description
          "The interface does not pass any packets.";
      }
    }
    description
      "interface up or down state for IGMP or MLD protocol";
  }
} // interface-config-attributes-igmp-mld

grouping interface-state-attributes-mld {
  description
    "Per interface state attributes for MLD.";

  uses interface-state-attributes-igmp-mld;

  leaf querier {
    type inet:ipv6-address;
    description
      "The querier address in the subnet.";
  }
  leaf-list joined-group {
    if-feature intf-join-group;
    type inet:ipv6-address;
    description
      "The routers that joined this multicast group.";
  }

  list group {
    key "address";
    description
      "Multicast group membership information
       that joined on the interface.";
  }
}
```

```
leaf address {
  type inet:ipv6-address;
  description
    "Multicast group address.";
}
uses interface-state-group-attributes-igmp-mld;
leaf-list host {
  type inet:ipv6-address;
  description
    "List of host address that
    joined the multicast group";
}
leaf last-reporter {
  type inet:ipv6-address;
  description
    "The last host address which has sent the
    report to join the multicast group.";
}
list source {
  key "address";
  description
    "List of multicast source information
    of the multicast group.";

  leaf address {
    type inet:ipv6-address;
    description
      "Multicast source address";
  }
  uses interface-state-source-attributes-igmp-mld;
  leaf last-reporter {
    type inet:ipv6-address;
    description
      "The last host address which has sent the
      report to join the multicast source and group.";
  }
} // list source
} // list group
} // interface-state-attributes-mld

grouping interface-state-group-attributes-igmp-mld {
  description
    "Per interface state attributes for both IGMP and MLD
    groups.";

  leaf expire {
    type uint32;
    units seconds;
    description
```

```
        "The time left before multicast group timeout.";
    }
    leaf filter-mode {
        type enumeration {
            enum "include" {
                description
                    "In include mode, reception of packets sent
                     to the specified multicast address is requested
                     only from those IP source addresses listed in the
                     source-list parameter";
            }
            enum "exclude" {
                description
                    "In exclude mode, reception of packets sent
                     to the given multicast address is requested
                     from all IP source addresses except those
                     listed in the source-list parameter.";
            }
        }
        description
            "Filter mode for a multicast group,
             may be either include or exclude.";
    }
    leaf host-count {
        type uint32;
        description
            "The number of host address.";
    }
    leaf up-time {
        type uint32;
        units seconds;
        description
            "The time after the device created multicast group record.";
    }
} // interface-state-group-attributes-igmp-mld

grouping interface-state-source-attributes-igmp-mld {
    description
        "Per interface state attributes for both IGMP and MLD
         groups.";

    leaf expire {
        type uint32;
        units seconds;
        description
            "The time left before multicast group timeout.";
    }
    leaf up-time {
        type uint32;
```

```
        units seconds;
        description
            "The time after the device created multicast group record.";
    }
} // interface-state-source-attributes-igmp-mld

/*
 * Configuration data nodes
 */
augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol"
{
    description
        "IGMP augmentation to routing control plane protocol configuration.";

    container igmp {
        description
            "IGMP configuration data.";

        container global {
            description
                "Global attributes.";
            uses global-config-attributes;
        }

        container interfaces {
            description
                "Containing a list of interfaces.";

            uses interfaces-config-attributes {
                if-feature global-interface-config;
            }

            list interface {
                key "interface-name";
                description
                    "List of IGMP interfaces.";
                leaf interface-name {
                    type if:interface-ref;
                    must "/if:interfaces/if:interface[if:name = current()]/"
                        + "ip:ipv4" {
                        description
                            "The interface must have IPv4 enabled.";
                    }
                }
                description
                    "Reference to an entry in the global interface
                    list.";
            }
            uses interface-config-attributes-igmp {
                if-feature per-interface-config;
            }
        }
    }
}
```

```

    }
  } // interface
} // interfaces
} // igmp
} // augment

augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol"
{
  description
    "MLD augmentation to routing control plane protocol configuration.";

  container mld {
    description
      "MLD configuration data.";

    container global {
      description
        "Global attributes.";
      uses global-config-attributes;
    }

    container interfaces {
      description
        "Containing a list of interfaces.";

      uses interfaces-config-attributes {
        if-feature global-interface-config;
      }

      list interface {
        key "interface-name";
        description
          "List of MLD interfaces.";
        leaf interface-name {
          type if:interface-ref;
          must "/if:interfaces/if:interface[if:name = current()]/"
            + "ip:ipv6" {
            description
              "The interface must have IPv6 enabled.";
          }
        }
        description
          "Reference to an entry in the global interface
            list.";
      }
      uses interface-config-attributes-mld {
        if-feature per-interface-config;
      }
    } // interface
  } // interfaces
}

```

```
    } // mld
  } // augment

/*
 * Operational state data nodes
 */
augment "/rt:routing-state/rt:control-plane-protocols/rt:control-plane-pro
tocol"
{
  description
    "IGMP augmentation to routing control plane protocol state.";

  container igmp {
    description
      "IGMP configuration data.";

    container global {
      description
        "Global attributes.";
      uses global-config-attributes;
      uses global-state-attributes;
    }

    container interfaces {
      description
        "Containing a list of interfaces.";

      uses interfaces-config-attributes {
        if-feature global-interface-config;
      }

      list interface {
        key "interface-name";
        description
          "List of IGMP interfaces.";
        leaf interface-name {
          type if:interface-ref;
          must "/if:interfaces/if:interface[if:name = current()]/"
            + "ip:ipv4" {
            description
              "The interface must have IPv4 enabled.";
          }
          description
            "Reference to an entry in the global interface
            list.";
        }
        uses interface-config-attributes-igmp {
          if-feature per-interface-config;
        }
        uses interface-state-attributes-igmp;
      }
    }
  }
}
```

```

        } // interface
      } // interfaces
    } // igmp
  } // augment

  augment "/rt:routing-state/rt:control-plane-protocols/rt:control-plane-pro
tocol"
  {
    description
      "MLD augmentation to routing control plane protocol state.";

    container mld {
      description
        "MLD configuration data.";

      container global {
        description
          "Global attributes.";
        uses global-config-attributes;
        uses global-state-attributes;
      }

      container interfaces {
        description
          "Containing a list of interfaces.";

        uses interfaces-config-attributes {
          if-feature global-interface-config;
        }

        list interface {
          key "interface-name";
          description
            "List of MLD interfaces.";
          leaf interface-name {
            type if:interface-ref;
            must "/if:interfaces/if:interface[if:name = current()]/"
              + "ip:ipv6" {
              description
                "The interface must have IPv6 enabled.";
            }
          }
          description
            "Reference to an entry in the global interface
            list.";
        }
        uses interface-config-attributes-mld {
          if-feature per-interface-config;
        }
        uses interface-state-attributes-mld;
      } // interface
    }
  }

```



```
    } // interfaces
  } // mld
} // augment

/*
 * RPCs
 */
rpc clear-igmp-groups {
  if-feature rpc-clear-groups;
  description
    "Clears the specified IGMP cache tables.";

  input {
    leaf interface {
      type leafref {
        path "/rt:routing/rt:control-plane-protocols"
          + "/rt:control-plane-protocol/"
          + "igmp/interfaces/interface/"
          + "interface-name";
      }
      description
        "Name of the IGMP interface.
        If it is not specified, groups from all interfaces are
        cleared.";
    }
    leaf group {
      type inet:ipv4-address;
      description
        "Multicast group IPv4 address.
        If it is not specified, all IGMP group tables are
        cleared.";
    }
    leaf source {
      type inet:ipv4-address;
      description
        "Multicast source IPv4 address.
        If it is not specified, all IGMP source-group tables are
        cleared.";
    }
  }
} // rpc clear-igmp-groups

rpc clear-mld-groups {
  if-feature rpc-clear-groups;
  description
    "Clears the specified MLD cache tables.";

  input {
    leaf interface {
```

```
    type leafref {
      path "/rt:routing/rt:control-plane-protocols"
        + "/rt:control-plane-protocol/"
        + "mld/interfaces/interface/"
        + "interface-name";
    }
    description
      "Name of the MLD interface.
      If it is not specified, groups from all interfaces are
      cleared.";
  }
  leaf group {
    type inet:ipv6-address;
    description
      "Multicast group IPv6 address.
      If it is not specified, all MLD group tables are
      cleared.";
  }
  leaf source {
    type inet:ipv6-address;
    description
      "Multicast source IPv6 address.
      If it is not specified, all MLD source-group tables are
      cleared.";
  }
} // rpc clear-mld-groups

/*
 * Notifications
 */
}
```

<CODE ENDS>

5. Security Considerations

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

6. IANA Considerations

RFC Ed.: In this section, replace all occurrences of 'XXXX' with the actual RFC number (and remove this note).

This document registers the following namespace URIs in the IETF XML registry [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-igmp-mld

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

This document registers the following YANG modules in the YANG Module Names registry [RFC7950]:

name: ietf-igmp-mld

namespace: urn:ietf:params:xml:ns:yang:ietf-igmp-mld

prefix: igmp-mld

reference: RFC XXXX

7. References

7.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016
- [RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, November 2016

7.2. Informative References

- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.
- [RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.
- [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.
- [RFC4541] M. Christensen, K. Kimball and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, May 2006.
- [RFC4605] B. Fenner, H. He, B. Haberman, and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.
- [RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011
- [I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-12(work in progress), March 2017.

8. Acknowledgments

The authors would like to thank Steve Baillargeon, Hu Fangwei, Robert Kebler, Tanmoy Kundu, Liu Yisong, and Stig Venaas for their valuable contributions.

Authors' Addresses

Xufeng Liu
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

EMail: Xufeng_Liu@jabil.com

Feng Guo
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: guofeng@huawei.com

Mahesh Sivakumar
Cisco Systems, Inc.
510 McCarthy Boulevard
Milpitas, California 95035
USA

Email: masivaku@cisco.com

Pete McAllister
Metaswitch Networks
100 Church Street
Enfield EN2 6BQ
UK

EMail: pete.mcallister@metaswitch.com

Anish Peter
Individual

EMail: anish.ietf@gmail.com

PIM WG
Internet-Draft
Intended status: Standards Track
Expires: September 2, 2017

Xufeng. Liu
Jabil
Zheng. Zhang
ZTE Corporation
Anish. Peter
Individual contributor
Mahesh. Sivakumar
Cisco Systems
Feng. Guo
Huawei Technologies
Pete. McAllister
Metaswitch Networks
March 1, 2017

MSDP YANG Model
draft-ietf-pim-msdp-yang-00

Abstract

This document defines a YANG data model for the configuration and management of MSDP Protocol.

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 2, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Design of the Data Model	2
3. MSDP configuration	6
4. MSDP State	6
5. MSDP RPC	6
6. Notifications	7
7. MSDP YANG model	7
8. Contributors	22
9. Normative References	22
Authors' Addresses	22

1. Introduction

[RFC3618] introduces the protocol definition of MSDP. This document defines a YANG data model that can be used to configure and manage the MSDP protocol. The operational state data and statistics can also be retrieved by this model.

This model is designed to be used along with other multicast YANG models such as PIM, which are not covered in this document.

2. Design of the Data Model

This model imports and augments ietf-routing YANG model defined in [I-D.ietf-netmod-routing-cfg]. Both configuration branch and state branch of [I-D.ietf-netmod-routing-cfg] are augmented. The configuration branch covers global configuration attributes and per peer configuration attributes. The state branch includes global, per peer, and source-active information. The container "msdp" is the top level container in this data model. The presence of this container is expected to enable MSDP protocol functionality.

```

module: ietf-msdp
augment /rt:routing/rt:control-plane-protocols:
  +--rw msdp!
    +--rw global
      |   +--rw connect-source?   if:interface-ref
      |   +--rw default-peer! {global-default-peer}?
      |   |   +--rw peer-addr      leafref

```



```

| | +--rw prefix-policy?   string {global-default-peer-policy}?
+--rw originating-rp
| | +--rw interface?   if:interface-ref
+--rw sa-filter
| | +--rw in?         string
| | +--rw out?        string
+--rw sa-limit?        uint32 {global-sa-limit}?
+--rw ttl-threshold?   uint8
+--rw peers
+--rw peer* [address]
+--rw address          inet:ipv4-address
+--rw authentication
| +--rw (authentication-type)?
|   +--:(key-chain) {peer-key-chain}?
|   | +--rw key-chain?          key-chain:key-chain-ref
|   +--:(password) {peer-key-chain}?
|   | +--rw key?                string
|   +--rw (algorithm)?
|   | +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
|   | | +--rw hmac-sha-1-12?          empty
|   | +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
|   | | +--rw aes-cmac-prf-128?        empty
|   | +--:(md5)
|   | | +--rw md5?                    empty
|   | +--:(sha-1)
|   | | +--rw sha-1?                  empty
|   | +--:(hmac-sha-1)
|   | | +--rw hmac-sha-1?              empty
|   | +--:(hmac-sha-256)
|   | | +--rw hmac-sha-256?            empty
|   | +--:(hmac-sha-384)
|   | | +--rw hmac-sha-384?            empty
|   | +--:(hmac-sha-512)
|   | | +--rw hmac-sha-512?            empty
|   | +--:(clear-text) {clear-text}?
|   | | +--rw clear-text?              empty
|   | +--:(replay-protection-only) {replay-protection-only}?
|   | | +--rw replay-protection-only?  empty
+--rw enable?          boolean {peer-admin-enable}?
+--rw connect-source?  if:interface-ref
+--rw description?     string {peer-description}?
+--rw mesh-group?      string
+--rw peer-as?         string {peer-as}?
+--rw sa-filter
| | +--rw in?         string
| | +--rw out?        string
+--rw sa-limit?        uint32 {peer-sa-limit}?
+--rw timer

```

```

?      |   +--rw connect-retry-interval?   uint16 {peer-timer-connect-retry}
      |   |
      |   |   +--rw holdtime-interval?      uint16 {peer-timer-holdtime}?
      |   |   +--rw keepalive-interval?     uint16 {peer-timer-keepalive}?
      |   +--rw ttl-threshold?             uint8
augment /rt:routing-state/rt:control-plane-protocols:
  +--ro msdp!
    +--ro global
      +--ro connect-source?   if:interface-ref
      +--ro default-peer! {global-default-peer}?
      |   +--ro peer-addr      leafref
      |   +--ro prefix-policy? string {global-default-peer-policy}?
      +--ro originating-rp
      |   +--ro interface?   if:interface-ref
      +--ro sa-filter
      |   +--ro in?          string
      |   +--ro out?         string
      +--ro sa-limit?        uint32 {global-sa-limit}?
      +--ro ttl-threshold?   uint8
    +--ro peers
      +--ro peer* [address]
        +--ro address          inet:ipv4-address
        +--ro authentication
          +--ro (authentication-type)?
            +--:(key-chain) {peer-key-chain}?
            |   +--ro key-chain?          key-chain:key-chain-ref
            +--:(password) {peer-key-chain}?
            +--ro key?                    string
            +--ro (algorithm)?
              +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
              |   +--ro hmac-sha1-12?      empty
              +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
              |   +--ro aes-cmac-prf-128?   empty
              +--:(md5)
              |   +--ro md5?                empty
              +--:(sha-1)
              |   +--ro sha-1?              empty
              +--:(hmac-sha-1)
              |   +--ro hmac-sha-1?        empty
              +--:(hmac-sha-256)
              |   +--ro hmac-sha-256?      empty
              +--:(hmac-sha-384)
              |   +--ro hmac-sha-384?      empty
              +--:(hmac-sha-512)
              |   +--ro hmac-sha-512?      empty
              +--:(clear-text) {clear-text}?
              |   +--ro clear-text?        empty
              +--:(replay-protection-only) {replay-protection-only}?
              +--ro replay-protection-only? empty

```

```

|
|   +--ro enable?                boolean {peer-admin-enable}?
|   +--ro connect-source?        if:interface-ref
|   +--ro description?           string {peer-description}?
|   +--ro mesh-group?            string
|   +--ro peer-as?               string {peer-as}?
|   +--ro sa-filter
|   |   +--ro in?                string
|   |   +--ro out?               string
|   +--ro sa-limit?              uint32 {peer-sa-limit}?
|   +--ro timer
|   |   +--ro connect-retry-interval?  uint16 {peer-timer-connect-retry}
?
|   |   +--ro holdtime-interval?      uint16 {peer-timer-holdtime}?
|   |   +--ro keepalive-interval?     uint16 {peer-timer-keepalive}?
|   +--ro ttl-threshold?          uint8
|   +--ro session-state?          enumeration
|   +--ro elapsed-time?           uint32
|   +--ro connect-retry-expire?    uint32
|   +--ro hold-expire?            uint32
|   +--ro is-default-peer?        boolean
|   +--ro keepalive-expire?       uint32
|   +--ro reset-count?            uint32
|   +--ro statistics
|   |   +--ro discontinuity-time?     yang:date-and-time
|   |   +--ro error
|   |   |   +--ro rpf-failure?      uint32
|   |   +--ro queue
|   |   |   +--ro size-in?          uint32
|   |   |   +--ro size-out?        uint32
|   |   +--ro received
|   |   |   +--ro keepalive?        yang:counter64
|   |   |   +--ro notification?     yang:counter64
|   |   |   +--ro sa-message?       yang:counter64
|   |   |   +--ro sa-response?      yang:counter64
|   |   |   +--ro sa-request?       yang:counter64
|   |   |   +--ro total?            yang:counter64
|   |   +--ro sent
|   |   |   +--ro keepalive?        yang:counter64
|   |   |   +--ro notification?     yang:counter64
|   |   |   +--ro sa-message?       yang:counter64
|   |   |   +--ro sa-response?      yang:counter64
|   |   |   +--ro sa-request?       yang:counter64
|   |   |   +--ro total?            yang:counter64
+--ro sa-cache
|   +--ro entry* [group source-addr]
|   |   +--ro group                inet:ipv4-address
|   |   +--ro source-addr          union
|   |   +--ro origin-rp* [rp-address]
|   |   |   +--ro rp-address       inet:ip-address

```

```

        |   +--ro is-local-rp?      boolean
        |   +--ro sa-adv-expire?    uint32
    +--ro up-time?                  uint32
    +--ro expire?                  uint32
    +--ro holddown-interval?       uint32
    +--ro peer-learned-from?      inet:ipv4-address
    +--ro rpf-peer?               inet:ipv4-address
rpcs:
  +---x msdp-clear-peer
  |   +---w input
  |   +---w peer-address?      inet:ipv4-address
  +---x msdp-clear-sa-cache {rpc-clear-sa-cache}?
  |   +---w input
  |   +---w entry!
  |   |   +---w group          inet:ipv4-address
  |   |   +---w source-addr?   union
  |   +---w peer-address?      inet:ipv4-address
  |   +---w peer-as?           string

```

3. MSDP configuration

MSDP configurations require peer configurations. Several peers may be configured in a mesh-group. The Source-Active information may be filtered by peers.

The configuration modeling branch is composed of MSDP global and peer configurations. The two parts are the most important parts of MSDP.

Besides the fundamental features of MSDP protocol, several optional features are included in the model. These features help the control of MSDP protocol. The peer features and SA features make the deployment and control easier. The connection parameters can be used to control the TCP connection because MSDP protocol is based on TCP. The authentication features make the protocol more secure. The filter features allow operators to avoid the irrelevant information.

4. MSDP State

MSDP states are composed of MSDP global state, MSDP peer state, statistics information and Sa-cache information. The statistics information and Sa-cache information helps the operator to retrieve the protocol condition.

5. MSDP RPC

The part is used to define some useful and ordinary operations of protocol management.

6. Notifications

This part will be updated in later version.

7. MSDP YANG model

```
<CODE BEGINS> file "ietf-msdp@2016-10-18.yang"
module ietf-msdp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-msdp";
  prefix msdp;

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

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

  import ietf-routing {
    prefix "rt";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-ip {
    prefix "ip";
  }

  import ietf-key-chain {
    prefix "key-chain";
  }

  organization
    "IETF PIM( Protocols for IP Multicast ) Working Group";

  contact
    "WG Web:    <http://tools.ietf.org/wg/pim/>
    WG List:    <mailto:pim@ietf.org>
    WG Chair:   Stig Venaas
                <mailto:stig@venaas.com>
    WG Chair:   Mike McBride
                <mailto:mmcbride7@gmail.com>

    Editors:    ";
```

```
description
  "The module defines the YANG definitions for MSDP.";

revision 2016-10-18 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for MSDP.
     RFC 3618: Multicast Source Discovery Protocol (MSDP).
     RFC 4624: Multicast Source Discovery Protocol (MSDP) MIB";
}

/*
 * Features
 */
feature global-connect-source {
  description
    "Support configuration of global connect-source.";
}

feature global-default-peer {
  description
    "Support configuration of global default peer.";
}

feature global-default-peer-policy {
  description
    "Support configuration of global default peer.";
}

feature global-sa-filter {
  description
    "Support configuration of global SA filter.";
}

feature global-sa-limit {
  description
    "Support configuration of global limit on SA entries.";
}

feature global-ttl-threshold {
  description
    "Support configuration of global ttl-threshold.";
}

feature rpc-clear-sa-cache {
  description
    "Support the rpc to clear SA cache.";
```

```
}

feature peer-admin-enable {
  description
    "Support configuration of peer administrative enabling.";
}

feature peer-as {
  description
    "Support configuration of peer AS number.";
}

feature peer-connect-source {
  description
    "Support configuration of global connect-source.";
}

feature peer-description {
  description
    "Support configuration of peer description.";
}

feature peer-key-chain {
  description
    "Support configuration of peer key-chain.";
}

feature peer-password {
  description
    "Support configuration of peer key-chain.";
}

feature peer-sa-limit {
  description
    "Support configuration of per peer limit on SA entries.";
}

feature peer-timer-connect-retry {
  description
    "Support configuration of peer timer for connect-retry.";
}

feature peer-timer-keepalive {
  description
    "Support configuration of peer timer for keepalive.";
}

feature peer-timer-holdtime {
```

```
    description
      "Support configuration of peer timer for holdtime.";
  }

  /*
   * Groupings
   */
  grouping authentication-container {
    description
      "A container defining authentication attributes.";
    container authentication {
      description
        "A container defining authentication attributes.";
      choice authentication-type {
        case key-chain {
          if-feature peer-key-chain;
          leaf key-chain {
            type key-chain:key-chain-ref;
            description
              "Reference to a key-chain.";
          }
        }
        case password {
          if-feature peer-key-chain;
          leaf key {
            type string;
            description
              "This leaf describes the authentication key.";
          }
        }
      }
      /*uses key-chain:crypto-algorithm-types;*/ /*will be modified in next
version*/
    }
    description
      "Choice of authentication.";
  }
} // authentication-container

grouping connect-source {
  description "Attribute to configure connect-source.";
  leaf connect-source {
    type if:interface-ref;
    must "/if:interfaces/if:interface[if:name = current()]/"
      + "ip:ipv4" {
      description
        "The interface must have IPv4 enabled.";
    }
  }
  description
    "The interface is to be the source for the TCP connection.
```



```
        It is a reference to an entry in the global interface
        list.";
    }
} // connect-source

grouping global-config-attributes {
    description "Global MSDP configuration.";

    uses connect-source {
        if-feature global-connect-source;
    }
    container default-peer {
        if-feature global-default-peer;
        presence "";
        description
            "The default peer accepts all MSDP SA messages.
            A default peer is needed in topologies where MSDP peers do
            not coexist with BGP peers. The reverse path forwarding
            (RPF) check on SA messages can fail, and no SA messages are
            accepted. In these cases, you can configure the peer as a
            default peer and bypass RPF checks.";
        leaf peer-addr {
            type leafref {
                path "../../peers/peer/address";
            }
            mandatory true;
            description
                "Reference to a peer that is in the peer list.";
        }
        leaf prefix-policy {
            if-feature global-default-peer-policy;
            type string;
            description
                "If specified, only those SA entries whose RP is permitted
                in the prefix list are allowed;
                if not specified, all SA messages from the default peer
                are accepted.";
        }
    }
} // default-peer

container originating-rp {
    description
        "The container of originating-rp.";
    leaf interface {
        type if:interface-ref;
        must "/if:interfaces/if:interface[if:name = current()]/"
            + "ip:ipv4" {
            description
```

```
        "The interface must have IPv4 enabled.";
    }
    description
        "Reference to an entry in the global interface
        list.
        IP address of the interface is used in the RP field of an
        SA message entry. When Anycast RPs are used, all RPs use
        the same IP address. This parameter can be used to define
        a unique IP address for the RP of each MSDP peer.
        By default, the software uses the RP address of the
        local system.";
    }
} // originating-rp

uses sa-filter-container {
    if-feature global-sa-filter;
}
leaf sa-limit {
    if-feature global-sa-limit;
    type uint32;
    description
        "A limit on the number of SA entries accepted. By default,
        there is no limit.";
}
uses ttl-threshold {
    if-feature global-ttl-threshold;
}
} // global-config-attributes

grouping global-state-attributes {
    description "Global MSDP state attributes.";
} // global-state-attributes

grouping peer-config-attributes {
    description "Per peer configuration for MSDP.";

    uses authentication-container;
    leaf enable {
        if-feature peer-admin-enable;
        type boolean;
        description
            "true to enable peer;
            false to disable peer.";
    }
    uses connect-source {
        if-feature peer-connect-source;
    }
    leaf description {
```

```
    if-feature peer-description;
    type string;
    description
        "The peer description.";
}
leaf mesh-group {
    type string;
    description
        "Configure this peer to be a member of a mesh group";
}
leaf peer-as {
    if-feature peer-as;
    type string;
    description
        "Peer's autonomous system number (ASN).";
}
uses sa-filter-container;
leaf sa-limit {
    if-feature peer-sa-limit;
    type uint32;
    description
        "A limit on the number of SA entries accepted from this peer.
        By default, there is no limit.";
}
container timer {
    description "Timer attributes.";
    leaf connect-retry-interval {
        if-feature peer-timer-connect-retry;
        type uint16;
        units seconds;
        default 30;
        description "SHOULD be set to 30 seconds. ";
    }
    leaf holdtime-interval {
        if-feature peer-timer-holdtime;
        type uint16;
        units seconds;
        must ". > 3";
        default 75;
        description "The SA-Hold-Down-Period of this msdp peer.";
    }
    leaf keepalive-interval {
        if-feature peer-timer-keepalive;
        type uint16;
        units seconds;
        must ". > 1 and . < ../holdtime-interval";
        default 60;
        description "The keepalive timer of this msdp peer.";
    }
}
```

```
    }  
  } // timer  
  uses ttl-threshold;  
} // peer-config-attributes  
  
grouping peer-state-attributes {  
  description "Per peer state attributes for MSDP.";   
  
  leaf session-state {  
    type enumeration {  
      enum disabled {  
        description "Disabled.";   
      }  
      enum inactive {  
        description "Inactive.";   
      }  
      enum listen {  
        description "Listen.";   
      }  
      enum connecting {  
        description "Connecting.";   
      }  
      enum established {  
        description "Established.";   
      }  
    }  
    description  
      "Peer session state.";   
    reference  
      "RFC3618: Multicast Source Discovery Protocol (MSDP).";   
  }  
  leaf elapsed-time {  
    type uint32;  
    units seconds;  
    description "Elapsed time for being in a state.";   
  }  
  leaf connect-retry-expire {  
    type uint32;  
    units seconds;  
    description "Connect retry expire time of peer connection.";   
  }  
  leaf hold-expire {  
    type uint32;  
    units seconds;  
    description "Hold expire time of peer connection.";   
  }  
  leaf is-default-peer {  
    type boolean;  
  }
```

```
        description "If this peer is default peer.";
    }
    leaf keepalive-expire {
        type uint32;
        units seconds;
        description "Keepalive expire time of this peer.";
    }
    leaf reset-count {
        type uint32;
        description "The reset count of this peer.";
    }
    uses statistics-container;
} // peer-config-attributes

grouping sa-cache-state-attributes {
    description "SA cache state attributes for MSDP.";

    leaf up-time {
        type uint32;
        units seconds;
        description "The up time of this sa cache.";
    }
    leaf expire {
        type uint32;
        units seconds;
        description "If this cache has expired.";
    }
    leaf holddown-interval {
        type uint32;
        units seconds;
        description "Holddown timer value for SA forwarding.";
    }
    leaf peer-learned-from {
        type inet:ipv4-address;
        description
            "The address of peer that we learned this SA from .";
    }
    leaf rpf-peer {
        type inet:ipv4-address;
        description "RPF peer.";
    }
} // sa-cache-state-attributes

grouping sa-filter-container {
    description "A container defining SA filters.";
    container sa-filter {
        description
            "Specifies an access control list (ACL) to filter source
```

```
        active (SA) messages coming in to or going out of the
        peer.";
    leaf in {
        type string;
        description
            "Filters incoming SA messages only.";
    }
    leaf out {
        type string;
        description
            "Filters outgoing SA messages only.";
    }
} // sa-filter
} // sa-filter-container

grouping ttl-threshold {
    description "Attribute to configure TTL threshold.";
    leaf ttl-threshold {
        type uint8 {
            range 1..255;
        }
        description
            "Maximum number of hops data packets can traverse before
            being dropped.";
    }
} // sa-ttl-threshold

grouping statistics-container {
    description
        "A container defining statistics attributes.";
    container statistics {
        description "";
        leaf discontinuity-time {
            type yang:date-and-time;
            description
                "The time on the most recent occasion at which any one
                or more of the statistic counters suffered a
                discontinuity. If no such discontinuities have occurred
                since the last re-initialization of the local
                management subsystem, then this node contains the time
                the local management subsystem re-initialized itself.";
        }
    }
    container error {
        description "";
        uses statistics-error;
    }
    container queue {
        description "";
    }
}
```

```
        uses statistics-queue;
    }
    container received {
        description "";
        uses statistics-sent-received;
    }
    container sent {
        description "";
        uses statistics-sent-received;
    }
}
} // statistics-container

grouping statistics-error {
    description
        "A grouping defining error statistics
        attributes.";
    leaf rpf-failure {
        type uint32;
        description "";
    }
} // statistics-error

grouping statistics-queue {
    description
        "A grouping defining queue statistics
        attributes.";
    leaf size-in {
        type uint32;
        description
            "The size of the input queue.";
    }
    leaf size-out {
        type uint32;
        description
            "The size of the output queue.";
    }
} // statistics-queue

grouping statistics-sent-received {
    description
        "A grouping defining sent and received statistics
        attributes.";
    leaf keepalive {
        type yang:counter64;
        description
            "The number of keepalive messages.";
    }
}
```

```
    leaf notification {
      type yang:counter64;
      description
        "The number of notification messages.";
    }
    leaf sa-message {
      type yang:counter64;
      description
        "The number of SA messages.";
    }
    leaf sa-response {
      type yang:counter64;
      description
        "The number of SA response messages.";
    }
    leaf sa-request {
      type yang:counter64;
      description
        "The number of SA request messages.";
    }
    leaf total {
      type yang:counter64;
      description
        "The number of total messages.";
    }
  } // statistics-sent-received

/*
 * Configuration data nodes
 */
augment "/rt:routing/rt:control-plane-protocols" {
  description
    "MSDP augmentation to routing instance configuration.";

  container msdp {
    presence "Container for MSDP protocol.";
    description
      "MSDP configuration data.";

    container global {
      description
        "Global attributes.";
      uses global-config-attributes;
    }

    container peers {
      description
        "Containing a list of peers.";
    }
  }
}
```



```
    list peer {
      key "address";
      description
        "List of MSDP peers.";
      leaf address {
        type inet:ipv4-address;
        description
          "";
      }
      uses peer-config-attributes;
    } // peer
  } // peers
} // msdp
} // augment

/*
 * Operational state data nodes
 */
augment "/rt:routing-state/rt:control-plane-protocols" {
  description
    "MSDP augmentation to routing instance state.";

  container msdp {
    presence "Container for MSDP protocol.";
    description
      "MSDP state data.";

    container global {
      description
        "Global attributes.";
      uses global-config-attributes;
      uses global-state-attributes;
    }

    container peers {
      description
        "Containing a list of peers.";

      list peer {
        key "address";
        description
          "List of MSDP peers.";
        leaf address {
          type inet:ipv4-address;
          description
            "The address of peer";
        }
        uses peer-config-attributes;
      }
    }
  }
}
```

```
        uses peer-state-attributes;
    } // peer
} // peers

container sa-cache {
    description
        "The sa cache information.";
    list entry {
        key "group source-addr";
        description "";
        leaf group {
            type inet:ipv4-address;
            description "The group address of this sa cache.";
        }
        leaf source-addr {
            type union {
                type enumeration {
                    enum '*' {
                        description "The source addr of this sa cache.";
                    }
                }
            type inet:ipv4-address;
        }
        description "";
    }
    list origin-rp {
        key "rp-address";
        description
            "";
        leaf rp-address {
            type inet:ip-address;
            description "The rp address.";
        }
        leaf is-local-rp {
            type boolean;
            description "";
        }
        leaf sa-adv-expire {
            type uint32;
            units seconds;
            description
                "Periodic SA advertisement timer expiring time on
                a local RP.";
        }
    }
    uses sa-cache-state-attributes;
} // entry
} // sa-cache
```

```
    } // msdp
  } // augment

/*
 * RPCs
 */
rpc msdp-clear-peer {
  description
    "Clears the session to the peer.";
  input {
    leaf peer-address {
      type inet:ipv4-address;
      description
        "Address of peer to be cleared. If this is not provided
        then all peers are cleared.";
    }
  }
}

rpc msdp-clear-sa-cache {
  if-feature rpc-clear-sa-cache;
  description
    "Clears MSDP source active (SA) cache entries.";
  input {
    container entry {
      presence "";
      description
        "The SA cache (S,G) or (*,G) entry to be cleared. If this
        is not provided, all entries are cleared.";
      leaf group {
        type inet:ipv4-address;
        mandatory true;
        description "";
      }
      leaf source-addr {
        type union {
          type enumeration {
            enum '*' {
              description "";
            }
          }
          type inet:ipv4-address;
        }
        description "";
      }
    }
  } // s-g
  leaf peer-address {
    type inet:ipv4-address;
```

```
    description
      "Peer IP address from which MSDP SA cache entries have been
       learned. If this is not provided, entries learned from all
       peers are cleared.";
  }
  leaf peer-as {
    type string;
    description
      "ASN from which MSDP SA cache entries have been learned.
       If this is not provided, entries learned from all AS's
       are cleared.";
  }
}
}
}
<CODE ENDS>
```

8. Contributors

The authors would like to thank Yisong Liu (liuyisong@huawei.com), Benchong Xu (xu.benchong@zte.com.cn), Tanmoy Kundu (tanmoy.kundu@alcatel-lucent.com) for their valuable contributions.

9. Normative References

- [I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-25 (work in progress), November 2016.
- [RFC3618] Fenner, B., Ed. and D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)", RFC 3618, DOI 10.17487/RFC3618, October 2003, <<http://www.rfc-editor.org/info/rfc3618>>.
- [RFC4624] Fenner, B. and D. Thaler, "Multicast Source Discovery Protocol (MSDP) MIB", RFC 4624, DOI 10.17487/RFC4624, October 2006, <<http://www.rfc-editor.org/info/rfc4624>>.
- [RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <<http://www.rfc-editor.org/info/rfc6087>>.

Authors' Addresses

Xufeng Liu
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

Email: Xufeng_Liu@jabil.com

Zheng Zhang
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing
China

Email: zhang.zheng@zte.com.cn

Anish Peter
Individual contributor

Email: anish.ietf@gmail.com

Mahesh Sivakumar
Cisco Systems
510 McCarthy Boulevard
Milpitas, California
USA

Email: masivaku@cisco.com

Feng Guo
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: guofeng@huawei.com

Pete McAllister
Metaswitch Networks
100 Church Street
Enfield EN2 6BQ
UK

Email: pete.mcallister@metaswitch.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: August 4, 2018

IJ. Wijnands
S. Venaas
Cisco Systems, Inc.
M. Brig
Aegis BMD Program Office
A. Jonasson
Swedish Defence Material Administration (FMV)
January 31, 2018

PIM Flooding Mechanism and Source Discovery
draft-ietf-pim-source-discovery-bsr-12

Abstract

PIM Sparse-Mode (PIM-SM) uses a Rendezvous Point (RP) and shared trees to forward multicast packets from new sources. Once last hop routers receive packets from a new source, they may join the Shortest Path Tree for the source for optimal forwarding. This draft defines a new mechanism that provides a way to support PIM-SM without the need for PIM registers, RPs or shared trees. Multicast source information is flooded throughout the multicast domain using a new generic PIM flooding mechanism. This allows last hop routers to learn about new sources without receiving initial data packets.

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 August 4, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Conventions Used in This Document	4
1.2. Terminology	4
2. Testing and Deployment Experiences	4
3. A Generic PIM Flooding Mechanism	5
3.1. PFM Message Format	6
3.2. Administrative Boundaries	7
3.3. Originating PFM Messages	7
3.4. Processing PFM Messages	9
3.4.1. Initial Checks	9
3.4.2. Processing and Forwarding of PFM Messages	10
4. Distributing Source Group Mappings	10
4.1. Group Source Holdtime TLV	10
4.2. Originating Group Source Holdtime TLVs	11
4.3. Processing GSH TLVs	13
4.4. The First Packets and Bursty Sources	13
4.5. Resiliency to Network Partitioning	14
5. Configurable Parameters	14
6. Security Considerations	15
7. IANA Considerations	16
8. Acknowledgments	16
9. References	16
9.1. Normative References	16
9.2. Informative References	17
Authors' Addresses	17

1. Introduction

PIM Sparse-Mode (PIM-SM) [RFC7761] uses a Rendezvous Point (RP) and shared trees to forward multicast packets to Last Hop Routers (LHR). After the first packet is received by a LHR, the source of the multicast stream is learned and the Shortest Path Tree (SPT) can be joined. This draft defines a new mechanism that provides a way to support PIM-SM without the need for PIM registers, RPs or shared trees. Multicast source information is flooded throughout the multicast domain using a new generic PIM flooding mechanism. By

removing the need for RPs and shared trees, the PIM-SM procedures are simplified, improving router operations, management and making the protocol more robust. Also the data packets are only sent on the SPTs, providing optimal forwarding.

This mechanism has some similarities to PIM Dense Mode (PIM-DM) with its State-Refresh signaling [RFC3973], except that there is no initial flooding of data packets for new sources. It provides the traffic efficiency of PIM-SM, while being as easy to deploy as PIM-DM. The downside is that it cannot provide forwarding of initial packets from a new source, see Section 4.4. PIM-DM is very different from PIM-SM and not as mature, Experimental vs Internet Standard, and there are only a few implementations. The solution in this document consists of a lightweight source discovery mechanism on top of the Source-Specific Multicast (SSM) [RFC4607] parts of PIM-SM. It is feasible to implement only a subset of PIM-SM to provide SSM support, and in addition implement the mechanism in this draft to offer a source discovery mechanism for applications that do not provide their own source discovery.

This document defines a generic flooding mechanism for distributing information throughout a PIM domain. While the forwarding rules are largely similar to Bootstrap Router mechanism (BSR) [RFC5059], any router can originate information, and it allows for flooding of any kind of information. Each message contains one or more pieces of information encoded as TLVs (type, length and value). This document defines one TLV used for distributing information about active multicast sources. Other documents may define additional TLVs.

Note that this document is experimental. While the flooding mechanism is largely similar to BSR, there are some concerns about scale as there can be multiple routers distributing information, and potentially larger amount of data that needs to be processed and stored. Distributing knowledge of active sources in this way is new, and there are some concerns, mainly regarding potentially large amounts of source states that need to be distributed. While there has been some testing in the field, we need to learn more about the forwarding efficiency, both the amount of processing per router, and propagation delay, and the amount of state that can be distributed. In particular, how many active sources one can support without consuming too many resources. There are also parameters, see Section 5, that can be tuned regarding how frequently information is distributed, and it is not clear what parameters are useful for different types of networks.

1.1. 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].

1.2. Terminology

RP: Rendezvous Point

BSR: Bootstrap Router

RPF: Reverse Path Forwarding

SPT: Shortest Path Tree

FHR: First Hop Router, directly connected to the source

LHR: Last Hop Router, directly connected to the receiver

PFM: PIM Flooding Mechanism

PFM-SD: PFM Source Discovery

SG Mapping: Multicast source group mapping

2. Testing and Deployment Experiences

A prototype of this specification has been implemented and there has been some limited testing in the field. The prototype was tested in a network with low bandwidth radio links. The network has frequent topology changes, including frequent link or router failures. Previously existing mechanisms like PIM-SM and PIM-DM were tested.

With PIM-SM the existing RP election mechanisms were found to be too slow. With PIM-DM, issues were observed with new multicast sources starving low bandwidth links even when there are no receivers, in some cases such that there was no bandwidth left for prune messages.

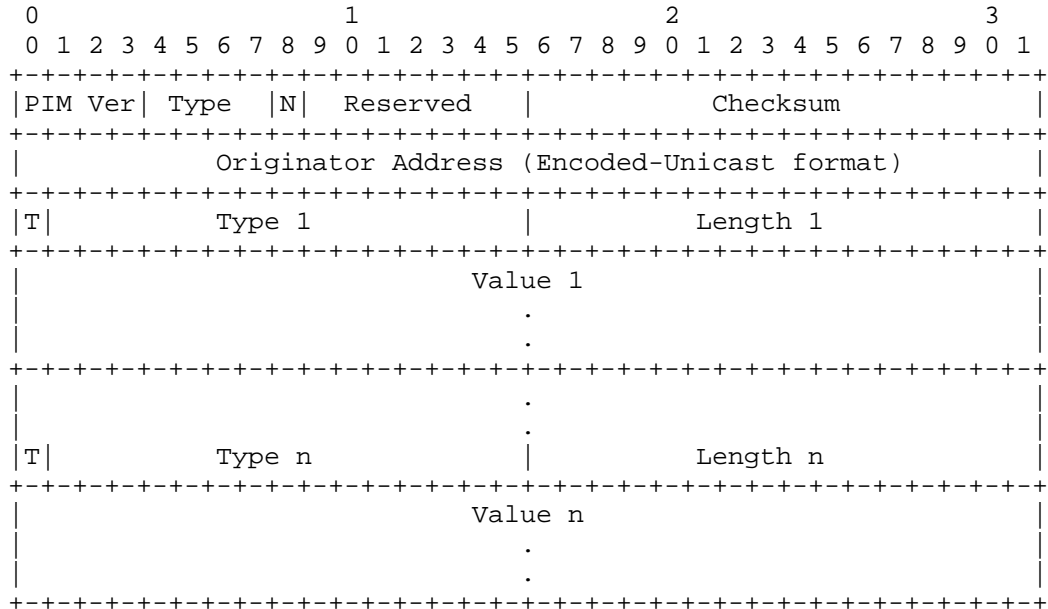
For the PFM-SD prototype tests, all routers were configured to send PFM-SD for directly connected source and to cache received announcements. Applications such as SIP with multicast subscriber discovery, multicast voice conferencing, position tracking and NTP were successfully tested. The tests went quite well. Packets were rerouted as needed and there were no unnecessary forwarding of packets. Ease of configuration was seen as a plus.

3. A Generic PIM Flooding Mechanism

The Bootstrap Router mechanism (BSR) [RFC5059] is a commonly used mechanism for distributing dynamic Group to RP mappings in PIM. It is responsible for flooding information about such mappings throughout a PIM domain, so that all routers in the domain can have the same information. BSR as defined, is only able to distribute Group to RP mappings. This document defines a more generic mechanism that can flood any kind of information. Administrative boundaries, see Section 3.2, may be configured to limit to which parts of a network the information is flooded.

The forwarding rules are identical to BSR, except that one can control whether routers should forward unsupported data types. For some types of information it is quite useful that it can be distributed without all routers having to support the particular type, while there may also be types where it is necessary for every single router to support it. The mechanism includes an originator address which is used for RPF checking to restrict the flooding, and prevent loops, just like BSR. Like BSR, messages are forwarded hop by hop; the messages are link-local and each router will process and resend the messages. Note that there is no equivalent to the BSR election mechanism; there can be multiple originators. This mechanism is named the PIM Flooding Mechanism (PFM).

3.1. PFM Message Format



PIM Version, Reserved and Checksum: As specified in [RFC7761].

Type: PIM Message Type. Value (pending IANA) for a PFM message.

[N]o-Forward bit: When set, this bit means that the PFM message is not to be forwarded. This bit is defined to prevent Bootstrap message forwarding in [RFC5059].

Originator Address: The address of the router that originated the message. This can be any address assigned to the originating router, but MUST be routable in the domain to allow successful forwarding. The format for this address is given in the Encoded-Unicast address in [RFC7761].

[T]ransitive bit: Each TLV in the message includes a bit called the Transitive bit that controls whether the TLV is forwarded by routers that do not support the given type. See Section 3.4.2.

Type 1..n: A message contains one or more TLVs, in this case n TLVs. The Type specifies what kind of information is in the Value. The type range is from 0 to 32767 (15 bits).

Length 1..n: The length of the the value field in octets.

Value 1..n: The value associated with the type and of the specified length.

3.2. Administrative Boundaries

PFM messages are generally forwarded hop by hop to all PIM routers. However, similar to BSR, one may configure administrative boundaries to limit the information to certain domains or parts of the network. Implementations MUST have a way of defining a set of interfaces on a router as administrative boundaries for all PFM messages, or optionally for certain TLVs, allowing for different boundaries for different TLVs. Usually one wants boundaries to be bidirectional, but an implementation MAY also provide unidirectional boundaries. When forwarding a message, a router MUST NOT send it out an interface that is an outgoing boundary, including bidirectional boundary, for all PFM messages. If an interface is an outgoing boundary for certain TLVs, the message MUST NOT be sent out the interface if it is a boundary for all the TLVs in the message. Otherwise the router MUST remove all the boundary TLVs from the message and send the message with the remaining TLVs. Also, when receiving a PFM message on an interface, the message MUST be discarded if the interface is an incoming boundary, including bidirectional boundary, for all PFM messages. If the interface is an incoming boundary for certain TLVs, the router MUST ignore all boundary TLVs. If all the TLVs in the message are boundary TLVs, then the message is effectively ignored. Note that when forwarding an incoming message, the boundary is applied before forwarding. If the message was discarded or all the TLVs were ignored, then no message is forwarded. When a message is forwarded, it MUST NOT contain any TLVs for which the incoming interface is an incoming, or bidirectional, boundary.

3.3. Originating PFM Messages

A router originates a PFM message when it needs to distribute information using a PFM message to other routers in the network. When a message is originated depends on what information is distributed. For instance this document defines a TLV to distribute information about active sources. When a router has a new active source, a PFM message should be sent as soon as possible. Hence a PFM message should be sent every time there is a new active source. However, the TLV also contains a holdtime and PFM messages need to be sent periodically. Generally speaking, a PFM message would typically be sent when there is a local state change, causing information to be distributed with PFM to change. Also, some information may need to be sent periodically. These messages are called triggered and periodic messages, respectively. Each TLV definition will need to define when a triggered PFM message needs to be originated, and also whether to send periodic messages, and how frequent.

A router MUST NOT originate more than `Max_PFM_Message_Rate` messages per minute. This document does not mandate how this should be implemented, but some possible ways could be having a minimal time between each message, counting the number of messages originated and resetting the count every minute, or using a leaky bucket algorithm. One benefit of using a leaky bucket algorithm is that it can handle bursts better. The default value of `Max_PFM_Message_Rate` is 6. The value MUST be configurable. Depending on the network, one may want to use a larger value of `Max_PFM_Message_Rate` to favor propagation of new information, but with a large number of routers and many updates, the total number of messages might become too large and require too much processing.

There MUST be a minimum of `Min_PFM_Message_Gap` milliseconds between each originated message. The default value of `Min_PFM_Message_Gap` is 1000 (1 second). The value MUST be configurable.

Unless otherwise specified by the TLV definitions, there is no relationship between different TLVs, and an implementation can choose whether to combine TLVs in one message or across separate messages. It is RECOMMENDED to combine multiple TLVs in one message, to reduce the number of messages, but it is also RECOMMENDED that the message is small enough to avoid fragmentation at the IP layer. When a triggered PFM message needs to be sent due to a state change, a router MAY send a message containing only the information that changed. If there are many changes occurring at about the same time, it might be possible to combine multiple changes in one message. In the case where periodic messages are also needed, an implementation MAY include periodic PFM information in a triggered PFM. E.g., if some information needs to be sent every 60 seconds and a triggered PFM is about to be sent 20 seconds before the next periodic PFM was scheduled, the triggered PFM might include the periodic information and the next periodic PFM can then be scheduled 60 seconds after that, rather than 20 seconds later.

When a router originates a PFM message, it puts one of its own addresses in the originator field. An implementation MUST allow an administrator to configure which address is used. For a message to be received by all routers in a domain, all the routers need to have a route for this address due to the RPF based forwarding. Hence an administrator needs to be careful which address to choose. When this is not configured, an implementation MUST NOT use a link-local address. It is RECOMMENDED to use an address of a virtual interface such that the originator can remain unchanged and routable independent of which physical interfaces or links may go down.

The No-Forward bit MUST NOT be set, except for the case when a router receives a PIM Hello from a new neighbor, or a PIM Hello with a new

Generation Identifier, defined in [RFC7761], is received from an existing neighbor. In that case an implementation MAY send PFM messages containing relevant information so that the neighbor can quickly get the correct state. The definition of the different PFM message TLVs need to specify what, if anything, needs to be sent in this case. If such a PFM message is sent, the No-Forward bit MUST be set, and the message must be sent within 60 seconds after the neighbor state change. The processing rules for PFM messages will ensure that any other neighbors on the same link ignores the message. This behavior and the choice of 60 seconds is similar to what is defined for the No-Forward bit in [RFC5059].

3.4. Processing PFM Messages

A router that receives a PFM message MUST perform the initial checks specified here. If the checks fail, the message MUST be dropped. An error MAY be logged, but otherwise the message MUST be dropped silently. If the checks pass, the contents is processed according to the processing rules of the included TLVs.

3.4.1. Initial Checks

In order to do further processing, a message MUST meet the following requirements. The message MUST be from a directly connected PIM neighbor, the destination address MUST be ALL-PIM-ROUTERS. Also, the interface MUST NOT be an incoming, nor bidirectional, administrative boundary for PFM messages, see Section 3.2. If No-Forward is not set, the message MUST be from the RPF neighbor of the originator address. If No-Forward is set, this system, the router doing these checks, MUST have enabled the PIM protocol within the last 60 seconds. See Section 3.3 for details. In pseudo-code the algorithm is as follows:

```

if ((DirectlyConnected(PFM.src_ip_address) == FALSE) OR
    (PFM.src_ip_address is not a PIM neighbor) OR
    (PFM.dst_ip_address != ALL-PIM-ROUTERS) OR
    (Incoming interface is admin boundary for PFM)) {
    drop the message silently, optionally log error.
}
if (PFM.no_forward_bit == 0) {
    if (PFM.src_ip_address !=
        RPF_neighbor(PFM.originator_ip_address)) {
        drop the message silently, optionally log error.
    }
} else if (more than 60 seconds elapsed since PIM enabled)) {
    drop the message silently, optionally log error.
}

```

Note that `src_ip_address` is the source address in the IP header of the PFM message. Originator is the originator field inside the PFM message, and is the router that originated the message. When the message is forwarded hop by hop, the originator address never changes, while the source address will be an address belonging to the router that last forwarded the message.

3.4.2. Processing and Forwarding of PFM Messages

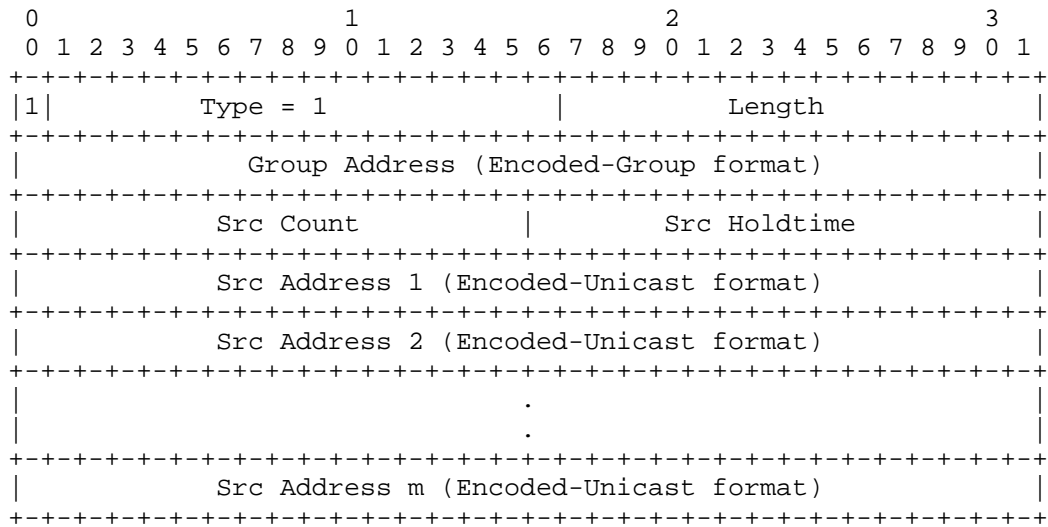
When the message is received, the initial checks above must be performed. If it passes the checks, then for each included TLV, perform processing according to the specification for that TLV.

After processing, the message is forwarded. Some TLVs may be omitted or modified in the forwarded message. This depends on administrative boundaries, see Section 3.2, the type specification and the setting of the Transitive bit for the TLV. If a router supports the type, then the TLV is forwarded with no changes unless otherwise specified by the type specification. A router not supporting the given type MUST include the TLV in the forwarded message if and only if the Transitive bit is set. Whether a router supports the type or not, the value of the Transitive bit MUST be preserved if the TLV is included in the forwarded message. The message is forwarded out of all interfaces with PIM neighbors (including the interface it was received on). As specified in Section 3.2, if an interface is an outgoing boundary for any TLVs, the message MUST NOT be sent out the interface if it is an outgoing boundary for all the TLVs in the message. Otherwise the router MUST remove any outgoing boundary TLVs of the interface from the message and send the message out that interface with the remaining TLVs.

4. Distributing Source Group Mappings

The generic flooding mechanism (PFM) defined in the previous section can be used for distributing source group mappings about active multicast sources throughout a PIM domain. A Group Source Holdtime (GSH) TLV is defined for this purpose.

4.1. Group Source Holdtime TLV



1: The Transitive bit is set to 1. This means that this type will be forwarded even if a router does not support it. See Section 3.4.2.

Type: This TLV has type 1.

Length: The length of the value in octets.

Group Address: The group that sources are to be announced for. The format for this address is given in the Encoded-Group format in [RFC7761].

Src Count: The number of source addresses that are included.

Src Holdtime: The Holdtime (in seconds) for the included source(s).

Src Address: The source address for the corresponding group. The format for these addresses is given in the Encoded-Unicast address in [RFC7761].

4.2. Originating Group Source Holdtime TLVs

A PFM message MAY contain one or more Group Source Holdtime (GSH) TLVs. This is used to flood information about active multicast sources. Each FHR that is directly connected to an active multicast source originates PFM messages containing GSH TLVs. How a multicast router discovers the source of the multicast packet and when it considers itself the FHR follows the same procedures as the registering process described in [RFC7761]. When a FHR has decided

that a register needs to be sent per [RFC7761], the SG is not registered via the PIM-SM register procedures, but the SG mapping is included in an GSH TLV in a PFM message. Note, only the SG mapping is distributed in the message, not the entire packet as would have been done with a PIM register.

The PFM messages containing the GSH TLV are sent periodically for as long as the multicast source is active, similar to how PIM registers are sent periodically. This means that as long as the source is active, it is included in a PFM message originated every Group_Source_Holdtime_Period seconds, within the general PFM timing requirements in Section 3.3. The default value of Group_Source_Holdtime_Period is 60. The value MUST be configurable. The holdtime for the source MUST be set to either zero or Group_Source_Holdtime_Holdtime. The value of the Group_Source_Holdtime_Holdtime parameter MUST be larger than Group_Source_Holdtime_Period. It is RECOMMENDED to be 3.5 times the Group_Source_Holdtime_Period. The default value is 210 (seconds). The value MUST be configurable. A source MAY be announced with a holdtime of zero to indicate that the source is no longer active.

If an implementation supports originating GSH TLVs with different holdtimes for different sources, it can if needed send multiple TLVs with the same group address. Due to the format, all the sources in the same TLV have the same holdtime.

When a new source is detected, an implementation MAY send a PFM message containing just that particular source. However, it MAY also include information about other sources that were just detected, sources that are scheduled for periodic announcement later, or other types of information. See Section 3.3 for details. Note that when a new source is detected, one should trigger sending of a PFM message as soon as possible, while if a source becomes inactive, there is no reason to trigger a message. There is no urgency in removing state for inactive sources. Note that the message timing requirements in Section 3.3 apply. This means that one cannot always send a triggered message immediately when a new source is detected. In order to meet the timing requirements, sending of the message may have to be delayed a small amount of time.

When a new PIM neighbor is detected, or an existing neighbor changes Generation Identifier, an implementation MAY send a triggered PFM message containing GSH TLVs for any Source Group mappings it has learned by receiving PFM GSH TLVs as well as any active directly connected sources. See Section 3.3 for further details.

4.3. Processing GSH TLVs

A router that receives a PFM message containing GSH TLVs MUST parse the GSH TLVs and store each of the GSH TLVs as SG mappings with a holdtimer started with the advertised holdtime, unless the implementation specifically does not support GSH TLVs, the router is configured to ignore GSH TLVs in general, or to ignore GSH TLVs for certain sources or groups. In particular, an administrator might configure a router to not process GSH TLVs if the router is known to never have any directly connected receivers.

For each group that has directly connected receivers, this router SHOULD send PIM (S,G) joins for all the SG mappings advertised in the message for the group. Generally joins are sent, but there could for instance be administrative policy limiting which sources and groups to join. The SG mappings are kept alive for as long as the holdtimer for the source is running. Once the holdtimer expires a PIM router MAY send a PIM (S,G) prune to remove itself from the tree. However, when this happens, there should be no more packets sent by the source, so it may be desirable to allow the state to time out rather than sending a prune.

Note that a holdtime of zero has a special meaning. It is to be treated as if the source just expired, and state to be removed. Source information MUST NOT be removed due to the source being omitted in a message. For instance, if there is a large number of sources for a group, there may be multiple PFM messages, each message containing a different list of sources for the group.

4.4. The First Packets and Bursty Sources

The PIM register procedure is designed to deliver Multicast packets to the RP in the absence of a Shortest Path Tree (SPT) from the RP to the source. The register packets received on the RP are decapsulated and forwarded down the shared tree to the LHRs. As soon as an SPT is built, multicast packets would flow natively over the SPT to the RP or LHR and the register process would stop. The PIM register process ensures packet delivery until an SPT is in place reaching the FHR. If the packets were not unicast encapsulated to the RP they would be dropped by the FHR until the SPT is setup. This functionality is important for applications where the initial packet(s) must be received for the application to work correctly. Another reason would be for bursty sources. If the application sends out a multicast packet every 4 minutes (or longer), the SPT is torn down (typically after 3:30 minutes of inactivity) before the next packet is forwarded down the tree. This will cause no multicast packet to ever be forwarded. A well behaved application should be able to deal with

packet loss since IP is a best effort based packet delivery system. But in reality this is not always the case.

With the procedures defined in this document the packet(s) received by the FHR will be dropped until the LHR has learned about the source and the SPT is built. That means for bursty sources or applications sensitive for the delivery of the first packet this solution would not be very applicable. This solution is mostly useful for applications that don't have strong dependency on the initial packet(s) and have a fairly constant data rate, like video distribution for example. For applications with strong dependency on the initial packet(s) using PIM Bidir [RFC5015] or SSM [RFC4607] is recommended. The protocol operations are much simpler compared to PIM SM, it will cause less churn in the network and both guarantee best effort delivery for the initial packet(s).

4.5. Resiliency to Network Partitioning

In a PIM SM deployment where the network becomes partitioned, due to link or node failure, it is possible that the RP becomes unreachable to a certain part of the network. New sources that become active in that partition will not be able to register to the RP and receivers within that partition are not able to receive the traffic. Ideally you would want to have a candidate RP in each partition, but you never know in advance which routers will form a partitioned network. In order to be fully resilient, each router in the network may end up being a candidate RP. This would increase the operational complexity of the network.

The solution described in this document does not suffer from that problem. If a network becomes partitioned and new sources become active, the receivers in that partitioned will receive the SG Mappings and join the source tree. Each partition works independently of the other partition(s) and will continue to have access to sources within that partition. Once the network has healed, the periodic flooding of SG Mappings ensures that they are re-flooded into the other partition(s) and other receivers can join to the newly learned sources.

5. Configurable Parameters

This document contains a number of configurable parameters. These parameters are formally defined in Section 3.3 and Section 4.2, but they are repeated here for ease of reference. These parameters all have default values as noted below.

Max_PFM_Message_Rate: The maximum number of PFM messages a router is allowed to originate per minute, see Section 3.3 for details. The default value is 6.

Min_PFM_Message_Gap: The minimum amount of time between each PFM message originated by a router in milliseconds, see Section 3.3 for details. The default is 1000.

Group_Source_Holdtime_Period: The announcement period for Group Source Holdtime TLVs in seconds, see Section 4.2 for details. The default value is 60.

Group_Source_Holdtime_Holdtime: The holdtime for Group Source Holdtime TLVs in seconds, see Section 4.2 for details. The default value is 210.

6. Security Considerations

When it comes to general PIM message security, see [RFC7761]. PFM messages **MUST** only be accepted from a PIM neighbor, but as discussed in [RFC7761], any router can become a PIM neighbor by sending a Hello message. To control from where to accept PFM packets, one can limit which interfaces PIM is enabled, and also one can configure interfaces as administrative boundaries for PFM messages, see Section 3.2. The implications of forged PFM messages depend on which TLVs they contain. Documents defining new TLVs will need to discuss the security considerations for the specific TLVs. In general though, the PFM messages are flooded within the network, and by forging a large number of PFM messages one might stress all the routers in the network.

If an attacker can forge PFM messages, then such messages may contain arbitrary GSH TLVs. An issue here is that an attacker might send such TLVs for a huge amount of sources, potentially causing every router in the network to store huge amounts of source state. Also, if there is receiver interest for the groups specified in the GSH TLVs, routers with directly connected receivers will build Shortest Path Trees for the announced sources, even if the sources are not actually active. Building such trees will consume additional resources on routers that the trees pass through.

PIM-SM link-local messages can be authenticated using IPsec, see [RFC7761] section 6.3 and [RFC5796]. Since PFM messages are link-local messages sent hop by hop, a link-local PFM message can be authenticated using IPsec such that a router can verify that a message was sent by a trusted neighbor and has not been modified. However, to verify that a received message contains correct information announced by the originator specified in the message, one

will have to trust every router on the path from the originator and that each router has authenticated the received message.

7. IANA Considerations

This document requires the assignment of a new PIM message type for the PIM Flooding Mechanism (PFM) with the name "PIM Flooding Mechanism". IANA is also requested to create a registry for PFM TLVs called "PIM Flooding Mechanism Message Types". Assignments for the registry are to be made according to the policy "IETF Review" as defined in [RFC8126]. The initial content of the registry should be:

Type	Name	Reference
0	Reserved	[this document]
1	Source Group Holdtime	[this document]
2-32767	Unassigned	

8. Acknowledgments

The authors would like to thank Arjen Boers for contributing to the initial idea, and David Black, Stewart Bryant, Yiqun Cai, Papadimitriou Dimitri, Toerless Eckert, Dino Farinacci, Alvaro Retana and Liang Xia for their very helpful comments on the draft.

9. References

9.1. 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>>.
- [RFC5059] Bhaskar, N., Gall, A., Lingard, J., and S. Venaas, "Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)", RFC 5059, DOI 10.17487/RFC5059, January 2008, <<https://www.rfc-editor.org/info/rfc5059>>.
- [RFC5796] Atwood, W., Islam, S., and M. Siami, "Authentication and Confidentiality in Protocol Independent Multicast Sparse Mode (PIM-SM) Link-Local Messages", RFC 5796, DOI 10.17487/RFC5796, March 2010, <<https://www.rfc-editor.org/info/rfc5796>>.

- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.

9.2. Informative References

- [RFC3973] Adams, A., Nicholas, J., and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", RFC 3973, DOI 10.17487/RFC3973, January 2005, <<https://www.rfc-editor.org/info/rfc3973>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <<https://www.rfc-editor.org/info/rfc5015>>.

Authors' Addresses

IJsbrand Wijnands
Cisco Systems, Inc.
De kleetlaan 6a
Diegem 1831
Belgium

Email: ice@cisco.com

Stig Venaas
Cisco Systems, Inc.
Tasman Drive
San Jose CA 95134
USA

Email: stig@cisco.com

Michael Brig
Aegis BMD Program Office
17211 Avenue D, Suite 160
Dahlgren VA 22448-5148
USA

Email: michael.brig@mda.mil

Anders Jonasson
Swedish Defence Material Administration (FMV)
Loennvaegen 4
Vaexjoe 35243
Sweden

Email: anders@jomac.se

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: January 1, 2018

A. Gupta
Avi Networks
S. Venaas
Cisco Systems
June 30, 2017

PIM Encoding and Procedures for Unicast IPv4 prefixes with IPv6 next-hop
draft-pim-with-ipv4-prefix-over-ipv6-nh-01.txt

Abstract

Multi-Protocol BGP (MP-BGP) has support for distributing next-hop information for multiple address families using one AFI/SAFI Network Layer Reachability Information (NLRI). [RFC5549] specifies the extensions necessary to allow advertising IPv4 NLRI or VPN-IPv4 NLRI with a Next Hop address that belongs to the IPv6 protocol. While the next-hop info is learnt via MP-BGP, certain procedures are needed to enable traffic forwarding. This document describes PIM extensions and the use-cases for multicast forwarding in various scenarios.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 1, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

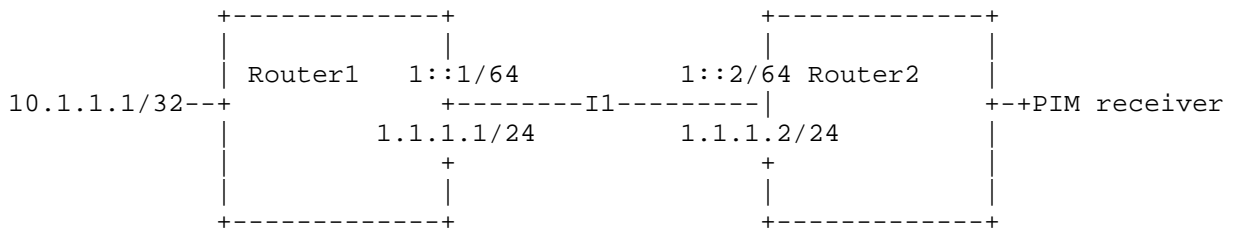
This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	2
2. Solution	3
3. Security Considerations	4
4. IANA Considerations	4
5. References	4
5.1. Normative References	4
5.2. Informative References	4
Authors' Addresses	4

1. Introduction

Figure 1: Example Topology



While use of MP-BGP along with [RFC5549] enables one routing protocol session to exchange next-hop info for both IPv4 and IPv6 prefixes, forwarding plane needs additional procedures to enable forwarding in

data-plane. For example, when a IPv4 prefix is learnt over IPv6 next-hop, forwarding plane resolves the MAC-Address (L2-Adjacency) for IPv6 next-hop and uses it as destination-mac while doing inter-subnet forwarding. While it's simple to find the required information for unicast forwarding, multicast forwarding in same scenario poses additional requirements.

Multicast traffic is forwarding on a tree build by multicast routing protocols such as PIM. Multicast routing protocols are address family dependent and hence a system enabled with IPv4 and IPv6 multicast routing will have two PIM sessions one for each of the AF. Also, Multicast routing protocol uses Unicast reachability information to find unique Reverse Path Forwarding Neighbor. Further it sends control messages such as PIM Join to form the tree. Now when a PIMv4 session needs to initiate new multicast tree in event of discovering new receiver It consults Unicast control plane to find next-hop information. While this multicast tree can be Shared or Shortest Path tree, PIMv4 will need a PIMv4 neighbor to send join. However, the Unicast control plane can provide IPv6 next-hop as explained earlier and hence we need certain procedures to find corresponding PIMv4 neighbor address. This address is vital for correct prorogation of join and furthermore to build multicast tree. This document describes various approaches along with their use-cases and pros-cons.

In example topology, Router1 and Router2 are PIMv4 and PIMv6 neighbors on Interface I1. Router2 learns prefix 10.1.1.1/32's next-hop as 1::164 on Interface I1 as advertised by Router1 using BGP IPV6 NLRI. But in order to send (10.1.1.1/32, multicast-group) PIMv4 join on Interface I1, Router1 needs to find corresponding PIMv4 neighbor. In case there are multiple PIMv4 neighbors on same Interface I1, problem is aggravated.

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 BCP 14, RFC 2119 [RFC2119].

2. Solution

A PIM router can advertise its locally configured IPv6 addresses on the interface in PIMv4 Hello messages as per RFC4601 section 4.3.4. Same applies for IPv4 address in PIMv6 Hello. PIM will keep this info for each neighbor in Neighbor-cache along with DR-priority, hold-time etc. Once IPv6 Next-hop is notified to PIMv4, it will look into neighbors on the notified RPF-interface and find PIMv4 neighbor advertising same IPv6 local address in secondary Neighbor-list. If

such a match is found, that particular neighbor will be used as IPv4 RPF-Neighbor for initiating upstream join.

This method is valid for networks enabled with PIMv4 and PIMv6 both as well for the networks enabled with only PIMv4 with IPv6 BGP session or PIMv6 with IPv4 BGP session. This method does't require any additional config changes in the network.

3. Security Considerations

There are no new security considerations.

4. IANA Considerations

There are no IANA considerations.

5. References

5.1. Normative References

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

5.2. Informative References

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <<http://www.rfc-editor.org/info/rfc4601>>.

[RFC5549] Le Faucheur, F. and E. Rosen, "Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop", RFC 5549, DOI 10.17487/RFC5549, May 2009, <<http://www.rfc-editor.org/info/rfc5549>>.

[RFC6395] Gulrajani, S. and S. Venaas, "An Interface Identifier (ID) Hello Option for PIM", RFC 6395, DOI 10.17487/RFC6395, October 2011, <<http://www.rfc-editor.org/info/rfc6395>>.

Authors' Addresses

Ashutosh Gupta
Avi Networks
5155 Old Ironsides Dr. Suite 100
Santa Clara, CA 95054
USA

Email: ashutosh@avinetworks.com

Stig Venaas
Cisco Systems
821 Alder Drive
San Jose, CA 95035
USA

Email: stig@cisco.com

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

Z. Zhang
Juniper Networks
K. Patel
Arrcus
I. Wijnands
Cisco Systems
A. Gulko
Thomson Reuters
March 13, 2017

BGP Based Multicast
draft-zzhang-bess-bgp-multicast-01

Abstract

This document specifies a BGP address family and related procedures that allow BGP to be used for setting up multicast distribution trees. This document also specifies procedures that enable BGP to be used for multicast source discovery, and for showing interest in receiving particular multicast flows. Taken together, these procedures allow BGP to be used as a replacement for other multicast routing protocols, such as PIM or mLDP. The BGP procedures specified here are based on the BGP multicast procedures that were originally designed for use by providers of Multicast Virtual Private Network service.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC2119.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 14, 2017.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Motivation	3
1.1.1. Native/unlabeled Multicast	3
1.1.2. Labeled Multicast	4
1.2. Overview	4
1.2.1. (x,g) Multicast	4
1.2.1.1. Source Discovery for ASM	5
1.2.1.2. ASM Shared-tree-only Mode	6
1.2.1.3. Integration with BGP-MVPN	6
1.2.2. BGP Inband Signaling for mLDP Tunnel	7
1.2.3. BGP Sessions	7
1.2.4. LAN and Parallel Links	8
1.2.5. Transition	9
2. Specification	9
2.1. BGP NLRIs and Attributes	9
2.1.1. S-PMSI A-D Route	10
2.1.2. Leaf A-D Route	11
2.1.3. Source Active A-D Route	12
2.1.4. S-PMSI A-D Route for C-multicast mLDP	12
2.1.5. Session Address Extended Community	12
2.2. Procedures	13
2.2.1. Source Discovery for ASM	13
2.2.2. Originating Tree Join Routes	13
2.2.2.1. (x,g) Multicast Tree	13
2.2.2.2. BGP Inband Signaling for mLDP Tunnel	14
2.2.3. Receiving Tree Join Routes	15
2.2.4. Withdrawl of Tree Join Routes	15
2.2.5. LAN procedures for (x,g) Unidirectional Tree	15
2.2.5.1. Originating S-PMSI A-D Routes	15

2.2.5.2. Receiving S-PMSI A-D Routes	16
2.2.6. Distributing Label for Upstream Traffic for Bidirectional Tree/Tunnel	17
3. Security Considerations	17
4. Acknowledgements	17
5. References	17
5.1. Normative References	17
5.2. Informative References	19
Authors' Addresses	19

1. Introduction

1.1. Motivation

This section provides some motivation for BGP signaling for native and labeled multicast. One target deployment would be a Data Center that requires multicast but uses BGP as its only routing protocol [RFC7938]. In such a deployment, it would be desirable to support multicast by extending the deployed routing protocol, without requiring the deployment of tree building protocols such as PIM, mLDP, RSVP-TE P2MP, and without requiring an IGP.

Additionally, compared to PIM, BGP based signaling has several advantage as described in the following section, and may be desired in non-DC deployment scenarios as well.

1.1.1. Native/unlabeled Multicast

Protocol Independent Multicast (PIM) has been the prevailing multicast protocol for many years. Despite its success, it has two drawbacks:

- o The ASM model, which is prevalent, introduces complexity in the following areas: source discovery procedures, need for Rendezvous Points (RPs) and group-to-RP mappings, need to switch between RP-rooted trees and source-rooted trees, etc.
- o Periodical protocol state refreshes due to soft state nature.

While PIM-SSM removes the complexity of PIM-ASM, it requires that multicast sources are known apriori. There have not been a good way of discovering sources, so its deployment has been limited. PIM-Port (PIM over Reliable Transport) solves the soft state issue, though its deployment has also been limited for two reasons:

- o It does not remove the ASM complexities.

- o In many of the scenarios where reliable transport is deemed important, BGP-based multicast (e.g. BGP-MVPN) has been used instead of PORT.

Partly because of the above mentioned problems, some Data Center operators have been avoiding deploying multicast in their networks.

BGP-MVPN [RFC6514] uses BGP to signal VPN customer multicast state over provider networks. It removes the above mentioned problems from the SP environment, and the deployment experiences have been encouraging. While RFC 6514 makes it possible for an SP to provide MVPN service without running PIM on its backbone, that RFC still assumes that PIM (or mLDP) runs on the PE-CE links. [draft-ietf-bess-mvpn-pe-ce] adapts the concept of BGP-MVPN to PE-CE links so that the use of PIM on the PE-CE links can be eliminated (though the PIM-ASM complexities still remains in the customer network), and this document extends it further to general topologies, so that they can be run on any router, as a replacement for PIM or mLDP.

With that, PIM can be completely eliminated from the network. PIM soft state is replaced by BGP hard state. For ASM, source specific trees are set up directly after simpler source discovery (data driven on FHRs and control driven elsewhere), all based on BGP. All the complexities related to source discovery and shared/source tree switch are also eliminated. Additionally, the trees can be setup with MPLS labels, with just minor enhancements in the signaling.

1.1.2. Labeled Multicast

There could be two forms of labeled multicast signaled by BGP. The first one is labeled (x,g) multicast where 'x' stands for either 's' or '*'. Basically, it is for BGP-signaled multicast tree as described in previous section but with labels. The second one is for mLDP tunnels with BGP signaling in part or whole through a BGP domain.

For both cases, BGP is used because other label distribution mechanisms like mLDP may not be desired by some operators. For example, a DC operator may prefer to have a BGP-only deployment.

1.2. Overview

1.2.1. (x,g) Multicast

PIM-like functionality is provided, using BGP-based join/prune signaling and BGP-based source discovery for ASM. The BGP-based join signaling supports both labeled multicast and IP multicast.

The same RPF procedures as in PIM are used for each router to determine the RPF neighbor for a particular source or RPA (in case of Bidirectional Tree). Except in the Bidirectional Tree case and a special case described in Section 1.2.1.2, no (*,G) join is used - LHR routers discover the sources for ASM and then join towards the sources directly. Data driven mechanisms like PIM Assert is replaced by control driven mechanisms (Section 1.2.4).

The joins are carried in BGP Updates with C-MCAST SAFI defined in [draft-ietf-bess-mvpn-pe-ce] and S-PMSI/Leaf A-D routes defined in this document. The updates are targeted at the upstream neighbor by use of Route Targets. [Note - earlier version of this draft uses C-multicast route to send joins. We're now switching to S-PMSI/Leaf routes for three reasons. a) when the routes go through RRs, we have to distinguish different routes based on upstream router and downstream router. This leads to Leaf routes. b) for labeled bidirectional trees, we need to signal "upstream fec". S-PMSI suits this very well. c) we may want to allow the option of setting up trees from the roots instead of from the leaves. S-PMSI suits that very well.]

If the BGP updates carry labels (via Tunnel Encapsulation Attribute [I-D.ietf-idr-tunnel-encaps]), then (s,g) multicast traffic can use the labels. This is very similar to mLDP Inband Signaling [RFC6826], except that there are no corresponding "mLDP tunnels" for the PIM trees. Similar to mLDP, labeled traffic on transit LANs are point to point. Of course, traffic sent to receivers on a LAN by a LHR is native multicast.

For labeled bidirectional (*,g) trees, downstream traffic (away from the RPA) can be forwarded as in the (s,g) case. For upstream traffic (towards RPA), the upstream neighbor needs to advertise a label for its downstream neighbors. The same label that the upstream neighbor advertises to its upstream is the same one that it advertises to its downstreams, using an S-PMSI A-D route.

1.2.1.1. Source Discovery for ASM

This document does not support ASM via shared trees (aka RP Tree, or RPT) with one exception discussed in the next section. Instead, FHRs, LHRs, and optionally RRs work together to propagate/discover source information via control plane and LHRs join source specific Shortest Path Trees (SPT) directly.

A FHR originates Source Active A-D routes upon discovering sources for particular flows and advertise them to its peers. It is desired that the SA routes only reach LHRs that are interested in receiving the traffic. To achieve that, the SA routes carry an IPv4 or IPv6

address specific Route Target. The Global Administrator field is set the group address of the flow, and the Local Administrator field is set to 0. An LHR advertises Route Target Membership routes, with the Route Target field in the NLRI set according to the groups it wants to receive traffic for, as how a FHR encode the Route Target in its Source Active routes. The propagation of the SA routes is subject to cooperative export filtering as specified in [RFC4684] and referred to as RTC mechanism in this document. That way, the LHR only receives Source Active routes for groups that it is interested in.

Typically, a set of RRs are used and they maintains all Source Active routes but only distribute to interested LHRs on demand (upon receiving corresponding Route Target Membership routes, which are triggered on LHRs when they receive IGMP/MLD membership routes). The rest of the document assumes that RRs are used, even though that is not required.

1.2.1.2. ASM Shared-tree-only Mode

It may be desired that only a shared tree is used to distribute all traffic for a particular ASM group from its RP to all LHRs, as described in Section 4.1 "PIM Shared Tree Forwarding" of [RFC7438]. This will significantly cut down the number of trees and works out very well in certain deployment scenarios. For example, all the sources could be connected to the RP, or clustered close to the RP. In the latter case, either the path from FHRs to the RP do not intersect the shared tree so native forwarding can be used between the FHRs and the RP, or other means outside of this document could be used to forward traffic from FHRs to the RP.

For native forwarding from FHRs to the RP, SA routes may be used to announce the sources so that the RP can join source specific trees to pull traffic, but the group specific Route Target is not needed. The LHRs do not advertise the group specific Route Target Membership routes as they do not need the SA routes.

To establish the shared tree, (*,g) Leaf A-D routes are used as in the bidirectional tree case, though no forwarding state is established to forward traffic from downstream neighbors.

1.2.1.3. Integration with BGP-MVPN

For each VPN, the Source Active routes distribution in that VPN do not have to involve PEs at all unless there are sources/receivers directly connected to some PEs and they are independent of MVPN SA routes. For example, FHRs and LHRs establish BGP sessions with RRs of that particular VPN for the purpose of SA distribution.

After source discovery, BGP multicast signaling is done from LHRs towards the sources. When the signaling reaches an egress PE, BGP-MVPN signaling takes over, as if a PIM (s,g) join/prune was received on the PE-CE interface. When the BGP-MVPN signaling reaches the ingress PE, BGP multicast signaling as specified in this document takes over, similar to how BGP-MVPN triggers PIM (s,g) join/prune on PE-CE interfaces.

1.2.2. BGP Inband Signaling for mLDP Tunnel

Part of an (or the whole) mLDP tunnel can also be signaled via BGP and seamlessly integrated with the rest of mLDP tunnel signaled natively via mLDP. All the procedures are similar to mLDP except that the signaling is done via BGP. The mLDP FEC is encoded as the BGP NLRI, with C-MCAST SAFI and S-PMSI/Leaf A-D Routes for C-multicast mLDP defined in this document. The Leaf A-D routes correspond to mLDP Label Mapping messages, and the S-PMSI A-D routes are used to signal upstream FEC for MP2MP mLDP tunnels, similar to the bidirection (*,g) case.

1.2.3. BGP Sessions

As specified in [draft-ietf-bess-mvpn-pe-ce-00], in order for two BGP speakers to exchange C-MCAST NLRI, they must use BGP Capabilities Advertisement [RFC5492] to ensure that they both are capable of properly processing the C-MCAST NLRI. This is done as specified in [RFC4760], by using a capability code 1 (multiprotocol BGP) with an AFI of IPv4 (1) or IPv6 (2) and a SAFI of C-MCAST with a value to be assigned by IANA.

How the BGP peer sessions are provisioned, whether EBGp or IBGP, whether statically, automatically (e.g., based on IGP neighbor discovery), or programmably via an external controller, is outside the scope of this document.

In case of IBGP, it could be that every router peering with Route Reflectors, or hop by hop IBGP sessions could be used to exchange C-MCAST NLRIs for joins. In the latter case, unless desired otherwise for reasons outside of the scope of this document, the hop by hop IBGP sessions SHOULD only be used to exchange C-MCAST NLRIs.

When multihop BGP is used, a router advertises its local interface addresses, for the same purposes that the Address List TLV in LDP serves. This is achieved by advertising the interface address as host prefixes with IPv4/v6 Address Specific ECs corresponding to the router's local addresses used for its BGP sessions (Section 2.1.5).

Because the BGP Capability Advertisement is only between two peers, when the sessions are only via RRs, a router needs another way to determine if its neighbor is capable of signaling multicast via BGP. The interface address advertisement can be used for that purpose - the inclusion of a Session Address EC indicates that the BGP speaker identified in the EC supports the C-Multicast NLRI.

FHRs and LHRs may also establish BGP sessions to some Route Reflectors for source discovery purpose (Section 1.2.1.1).

With the traditional PIM, the FHRs and LHRs refer to the PIM DRs on the source or receiver networks. With BGP based multicast, PIM may not be running at all, and the FHRs and LHRs refer to the IGMP/MLD queriers, or the DF elected per [I-D.wijnands-bier-mld-lan-election]. Alternatively, if it is known that a network only has senders then no IGMP/MLD or DF election is needed - any router may generate SA routes. That will not cause any issue other than redundant SA routes being originated.

1.2.4. LAN and Parallel Links

There could be parallel links between two BGP peers. A single multi-hop session, whether IBGP or EBGP, between loopback addresses may be used. Except for LAN interfaces in case of unlabeled (x,g) unidirectional trees (note that transit LAN interface is not supported for BGP signaled (*,g) bidirectional tree and for mLDp tunnels, traffic on transit LAN is point to point between neighbors), any link between the two peers can be automatically used by a downstream peer to receive traffic from the upstream peer, and it is for the upstream peer to decide which link to use. If one of the links goes down, the upstream peer switches to a different link and there is no change needed on the downstream peer.

For unlabeled (x,g) unidirectional trees, the upstream peer MAY prefer LAN interfaces to send traffic, since multiple downstream peers may be reached simultaneously, or it may make a decision based on local policy, e.g., for load balancing purpose. Because different downstream peers might choose different upstream peers for RPF, when an upstream peer decides to use a LAN interface to send traffic, it originates an S-PMSI A-D route indicating that one or more LAN interface will be used. The route carries Route Targets specific to the LANs so that all the peers on the LANs import the route. If more than one router originate the route specifying the same LAN for the same (s,g) or (*,g) flow, then assert procedure based on the S-PMSI A-D routes happens and assert losers will stop sending traffic to the LAN.

1.2.5. Transition

A network currently running PIM can be incrementally transitioned to BGP based multicast. At any time, a router supporting BGP based multicast can use PIM with some neighbors (upstream or downstream) and BGP with some other neighbors. PIM and BGP MUST not be used simultaneously between two neighbors for multicast purpose, and routers connected to the same LAN MUST be transitioned during the same maintenance window.

In case of PIM-SSM, any router can be transitioned at any time (except on a LAN all routers must be transitioned together). It may receive source tree joins from a mixed set of BGP and PIM downstream neighbors and send source tree joins to its upstream neighbor using either PIM or BGP signaling.

In case of PIM-ASM, the RPs are first upgraded to support BGP based multicast. They learn sources either via PIM procedures from PIM FHRs, or via Source Active A-D routes from BGP FHRs. In the former case, the RPs can originate proxy Source Active A-D routes. There may be a mixed set of RPs/RRs - some capable of both traditional PIM RP functionalities while some only redistribute SA routes.

Then any routers can be transitioned incrementally. A transitioned LHR router will pull Source Active A-D routes from the RPs/RRs when they receive IGMP/MLD (*,G) joins for ASM groups, and may send either PIM (s,g) joins or BGP Source Tree Join routes. A transitioned transit router may receive (*,g) PIM joins but only send source tree joins after pulling Source Active A-D routes from RPs/RRs.

Similarly, a network currently running mLDP can be incrementally transitioned to BGP signaling. Without the complication of ASM, any router can be transitioned at any time, even without the restriction of coordinated transition on a LAN. It may receive mixed mLDP label mapping or BGP updates from different downstream neighbors, and may exchange either mLDP label mapping or BGP updates with its upstream neighbors, depending on if the neighbor is using BGP based signaling or not.

2. Specification

2.1. BGP NLRIs and Attributes

The C-MCAST SAFI defined in [I-D.ietf-bess-mvpn-pe-ce] is used, but new route types are used as defined in this document.

- 3 - S-PMSI A-D Route for (x,g)
- 4 - Leaf A-D Route
- 5 - Source Active A-D Route
- 0x43 - S-PMSI A-D Route for C-multicast mLDP

Except for the Source Active A-D routes, the routes are to be consumed by targeted upstream/downstream neighbors, and are not propagated further. This can be achieved by outbound filtering based on the RTs that lead to the importation of the routes.

The Type-3/4 routes MAY carry a Tunnel Encapsulation Attribute (TEA) [I-D.ietf-idr-tunnel-encaps]. The Type-0x43 route MUST carry a TEA. When used for mLDP, the Type-4 route MUST carry a TEA. Only the MPLS tunnel type for the TEA is considered. Others are outside the scope of this document.

2.1.1. S-PMSI A-D Route

Similar to defined in RFC 6514, an S-PMSI A-D Route Type specific C-MCAST NLRI consists of the following, though it does not have an RD:

```

+-----+
| Multicast Source Length (1 octet) |
+-----+
| Multicast Source (variable)       |
+-----+
| Multicast Group Length (1 octet) |
+-----+
| Multicast Group (variable)        |
+-----+
| Upstream Router's IP Address      |
+-----+

```

If the Multicast Source (or Group) field contains an IPv4 address, then the value of the Multicast Source (or Group) Length field is 32. If the Multicast Source (or Group) field contains an IPv6 address, then the value of the Multicast Source (or Group) Length field is 128.

Usage of other values of the Multicast Source Length and Multicast Group Length fields is outside the scope of this document.

There are two usages for S-PMSI A-D route. They're described in Section 2.2.5 and Section 2.2.6 respectively.

2.1.2. Leaf A-D Route

Similar to the Leaf A-D route in [RFC6514], a C-MCAST Leaf A-D route's route key includes the corresponding S-PMSI NLRI, plus the Originating Router's IP Addr. The difference is that there is no RD.

+-----+	
	S-PMSI NLRI
+-----+	
	Originating Router's IP Address
+-----+	

For example, the entire NLRI of a Leaf A-D route for (x,g) tree is as following:

			+ -																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																													</
--	--	--	-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

Even though the C-MCAST Leaf A-D route is unsolicited, unlike the Leaf A-D route for GTM in [RFC7524], it is encoded as if a corresponding S-PMSI A-D route had been received.

When used for signaling mLDP tunnels, even though the Leaf A-D route is unsolicited, unlike the "Route-type 0x44 Leaf A-D route for C-multicast mLDP" as in [RFC7441], it is Route-type 4 and encoded as if a corresponding S-PMSI A-D route had been received.

2.1.3. Source Active A-D Route

Similar to defined in RFC 6514, a Source Active A-D Route Type specific MCAST NLRI consists of the following:

```

+-----+
| Multicast Source Length (1 octet) |
+-----+
|   Multicast Source (variable)   |
+-----+
| Multicast Group Length (1 octet) |
+-----+
|   Multicast Group (variable)   |
+-----+

```

The definition of the source/length and group/length fields are the same as in the S-PMSI A-D routes.

Usage of Source Active A-D routes is described in Section 1.2.1.1.

2.1.4. S-PMSI A-D Route for C-multicast mLDP

The route is used to signal upstream FEC for an MP2MP mLDP tunnel. The route key include the mLDP FEC and the Upstream Router's IP Address field. The encoding is similar to the same route in [RFC7441], though there is no RD.

2.1.5. Session Address Extended Community

For two BGP speakers to determine if they are directly connected, each will advertise their local interface addresses, with an Session Address Extended Community. This is an Address Specific EC, with the Global Admin Field set to the local address used for its multihop sessions and the Local Admin Field set to the prefix length corresponding to the interface's network mask.

For example, if a router has two interfaces with address 10.10.10.1/24 and 10.12.0.1/16 respectively (notice the different network mask), and a loopback address 11.11.11.1/32 that is used for BGP sessions, then it will advertise prefix 10.10.10.1/32 with a Session Address EC 11.11.11.1:24 and 10.12.0.1/32 with a Session Address EC 11.11.11.1:16. If it also uses another loopback address 11.11.11.11/32 for other BGP sessions, then the routes will additionally carry Session Address EC 11.11.11.11:24 and 11.11.11.11:16 respectively.

This achieves what the Address List TLV in LDP Address Messages achieves, and can also be used to indicate that a router supports the BGP multicast signaling procedures specified in this document.

Only those interface addresses that will be used as resolved nexthops in the RIB need to be advertised with the Session Address EC. For example, the RPF lookup may say that the resolved nexthop address is A1, so the router needs to find out the corresponding BGP speaker with address A1 through the (interface address, session address) mapping built according to the interface address NLRI with the Session Address EC. For comparison with LDP, this is done via the (interface address, session address) mapping that is built by the LDP Address Messages.

2.2. Procedures

2.2.1. Source Discovery for ASM

When a FHR first receives a multicast packet addressed to an ASM group, it originates a Source Active route. It carries a IP/IPv6 Address Specific RT, with the Global Admin Field set to the group address and the Local Admin Field set to 0. The route is advertised to its peers, who will re-advertise further based on the RTC mechanisms. Note that typically the route is advertised only to the RRs.

The FHRs withdraws the Source Active route after a certain amount of time since it last received a packet of an (s,g) flow. The amount of time to wait is a local matter.

2.2.2. Originating Tree Join Routes

Note that in this document, tree join routes are S-PMSI/Leaf A-D routes.

2.2.2.1. (x,g) Multicast Tree

When a router learns from IGMP/MLD or a downstream PIM/BGP peer that it needs to join a particular (s,g) tree, it determines the RPF nexthop address wrt the source, following the same RPF procedures as defined for PIM. It further finds the BGP router that advertised the nexthop address as one of its local addresses.

If the RPF neighbor supports C-MCAST SAFI, this router originates a Leaf A-D route. Although it is unsolicited, it is constructed as if there was a corresponding S-PMSI A-D route. The Upstream Router's IP Address field is set to the RPF neighbor's session address (learnt via the EC attached to the host route for the RPF nexthop address).

An Address Specific RT corresponding to the session address is attached to the route, with the Global Administrative Field set to the session address and the local administrative field set to 0.

Similarly, when a router learns that it needs to join a bi-directional tree for a particular group, it determines the RPF neighbor wrt the RPA. If the neighbor supports C-MCAST SAFI, it originates a Leaf A-D Route and advertises the route to the RPF neighbor (in case of EBGp or hop-by-hop IBGP), or one or more RRs.

When a router first learns that it needs to receive traffic for an ASM group, either because of a local (*,g) IGMP/MLD report or a downstream PIM (*,g) join, it originates a RTC route with the NLRI's AS field set to its AS number and the Route Target field set to an address based RT, with the Global Administrator field set to group address and the Local Administrator field set to 0. The route is advertised to its peers (most practically some RRs), so that the router can receive matching Source Active A-D routes. Upon the receiving of the Source Active A-D routes, the router originates Leaf A-D routes as described above, as long as it still needs to receive traffic for the flows (i.e., the corresponding IGMP/MLD membership exists or join from downstream PIM/BGP neighbor exists).

When a Leaf A-D route is originated by this router, it sets up corresponding forwarding state such that the expected incoming interface list includes all non-LAN interfaces directly connecting to the upstream neighbor. LAN interfaces are added upon receiving corresponding S-PMSI A-D route (Section 2.2.5.2). If the upstream neighbor is not directly connected, tunnels may be used - details to be included in future revisions.

When the upstream nbr changes, the previously advertised Leaf A-D route is withdrawn. If there is a new upstream neighbor, a new Leaf A-D route is originated, corresponding to the new neighbor. Because NLRIs are different for the old and new Leaf A-D routes, make-before-break can be achieved, so can MoFRR [RFC7431].

2.2.2.2. BGP Inband Signaling for mLDP Tunnel

The same mLDP procedures as defined in [RFC6388] are followed, except that where a label mapping message is sent in [RFC6388], a Leaf A-D route is sent if the the upstream neighbor supports BGP based signaling.

2.2.3. Receiving Tree Join Routes

A router (auto-)configures Import RTs matching itself so that it can import tree join routes from their peers. Note that in this document, tree join routes are S-PMSI/Leaf A-D routes.

When a router receives a tree join route and imports it, it determines if it needs to originate its own corresponding route and advertise further upstream wrt the source/RPA or mLDP tunnel root. If itself is the FHR or is on the RPL or is the tunnel root, then it does not need to. Otherwise the procedures in Section 2.2.2 are followed.

Additionally, the router sets up its corresponding forwarding state such that traffic will be sent to the downstream neighbor, and received from the downstream neighbor in case of bidirectional tree/tunnel. If the downstream neighbor is not directly connected, tunnels may be used - details to be included in future revisions.

2.2.4. Withdrawl of Tree Join Routes

For a particular tree or tunnel, if a downstream neighbor withdraw its Leaf A-D route, the neighbor is removed from the corresponding forwarding state. If all downstream neighbors withdraw their tree join routes and this router no longer has local receivers, it withdraws the tree join routes that it previously originated.

As mentioned earlier, when the upstream neighbor changes, the previously advertised Leaf A-D route is also withdrawn. The corresponding incoming interfaces are also removed from the corresponding forwarding state.

2.2.5. LAN procedures for (x,g) Unidirectional Tree

For a unidirectional (x,g) multicast tree, if there is a LAN interface connecting to the downstream neighbor, it MAY be preferred over non-LAN interfaces, but an S-PMSI A-D route MUST be originated to facilitate the analog of the Assert process (Section 2.2.5.1).

2.2.5.1. Originating S-PMSI A-D Routes

If this router chooses to use a LAN interface to send traffic to its neighbors for a particular (s,g) or (*,g) flow, it MUST announce that by originating a corresponding S-PMSI A-D route. The Tunnel Type in the PMSI Tunnel Attribute (PTA) is set to 0 (no tunnel information Present). The LAN interface is identified by an IP address specific RT, with the Global Administrative Field set to the LAN interface's address prefix and the Local Administrative Field set to the prefix

length. The RT also serves the purpose of restricting the importing of the route by all routers on the LAN. An operator MUST ensure that RTs encoded as above are not used for other purposes. Practically that should not be unreasonable.

If multiple LAN interfaces are to be used (to reach different sets of neighbors), then the route will include multiple RTs, one for each used LAN interface as described above.

The S-PMSI A-D routes may also be used to announce tunnels that could be used to send traffic to downstream neighbors that are not directly connected. Details may be added in future revisions.

2.2.5.2. Receiving S-PMSI A-D Routes

A router (auto-)configures an Import RT for each of its LAN interfaces over which BGP is used for multicast signaling. The construction of the RT is described in the previous section.

When a router R1 imports an S-PMSI A-D route for flow (x,g) from router R2, R1 checks to see if it also originates an S-PMSI A-D route with the same NLRI except the Upstream Router's IP Address field. When a router R1 originates an S-PMSI A-D route, it checks to see if it also has installed an S-PMSI A-D route, from some other router R2, with the same NLRI except the Upstream Router's IP Address field. In either case, R1 checks to see if the two routes have an RT in common and the RT is encoded as in Section 2.2.5.1. If so, then there is a LAN attached to both R1 and R2, and both routers are prepared to send (S,G) traffic onto that LAN. This kicks off the assert procedure to elect a winner - the one with the highest Upstream Router's IP Address in the NLRI wins. An assert loser will not include the corresponding LAN interface in its outgoing interface list, but it keeps the S-PMSI A-D route that it originates.

If this router does not have a matching S-PMSI route of its own with some common RTs, and the originator of the received S-PMSI route is a chosen upstream neighbor for the corresponding flow, then this router updates its forwarding state to include the LAN interface in the incoming interface list. When the last S-PMSI route with a RT matching the LAN is withdrawn later, the LAN interface is removed from the incoming interface list.

Note that a downstream router on the LAN does not participate in the assert procedure. It adds/keeps the LAN interface in the expected incoming interfaces as long as its chosen upstream peer originates the S-PMSI AD route. It does not switch to the assert winner as its upstream. An assert loser MAY keep sending joins upstream based on

local policy even if it has no other downstream neighbors (this could be used for fast switch over in case the assert winner would fail).

2.2.6. Distributing Label for Upstream Traffic for Bidirectional Tree/Tunnel

For MP2MP mLDP tunnels or labeled (*,g) bidirectional trees, an upstream router needs to advertise a label to all its downstream neighbors so that the downstream neighbors can send traffic to itself.

For MP2MP mLDP tunnels, the same procedures for mLDP are followed except that instead of MP2MP-U Label Mapping messages, S-PMSI A-D Routes for C-Multicast mLDP are used.

For labeled (*,g) bidirectional trees, for a Leaf A-D route received from a downstream neighbor, a corresponding S-PMSI A-D route is sent back to the downstream router.

In both cases, a single S-PMSI A-D route is originated for each tree from this router, but with multiple RTs (one for each downstream neighbor on the tree). A TEA specifies a label allocated by the upstream router for its downstream neighbors to send traffic with. Note that this is still a "downstream allocated" label (the upstream router is "downstream" from traffic direction point of view).

The S-PMSI routes do not carry a PTA, unless a P2MP tunnel is used to reach downstream neighbors. Such use case is out of scope of this document for now and may be specified in the future.

3. Security Considerations

This document does not introduce new security risks?

4. Acknowledgements

The authors thank Marco Rodrigues and Lenny Giuliano for their initial idea/ask of using BGP for multicast signaling beyond MVPN. We also thank Eric Rosen for his questions, suggestions, and help finding solutions to some issues.

5. References

5.1. Normative References

- [I-D.ietf-bess-mvpn-pe-ce]
Patel, K., Rosen, E., and Y. Rekhter, "BGP as an MVPN PE-CE Protocol", draft-ietf-bess-mvpn-pe-ce-01 (work in progress), October 2015.
- [I-D.ietf-idr-tunnel-encaps]
Rosen, E., Patel, K., and G. Velde, "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-03 (work in progress), November 2016.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <<http://www.rfc-editor.org/info/rfc4601>>.
- [RFC4684] Marques, P., Bonica, R., Fang, L., Martini, L., Raszuk, R., Patel, K., and J. Guichard, "Constrained Route Distribution for Border Gateway Protocol/MultiProtocol Label Switching (BGP/MPLS) Internet Protocol (IP) Virtual Private Networks (VPNs)", RFC 4684, DOI 10.17487/RFC4684, November 2006, <<http://www.rfc-editor.org/info/rfc4684>>.
- [RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <<http://www.rfc-editor.org/info/rfc5015>>.
- [RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <<http://www.rfc-editor.org/info/rfc6514>>.
- [RFC7441] Wijnands, IJ., Rosen, E., and U. Joerde, "Encoding Multipoint LDP (mLDP) Forwarding Equivalence Classes (FECs) in the NLRI of BGP MCAST-VPN Routes", RFC 7441, DOI 10.17487/RFC7441, January 2015, <<http://www.rfc-editor.org/info/rfc7441>>.

5.2. Informative References

- [I-D.wijnands-bier-mld-lan-election]
Wijnands, I., Pfister, P., and Z. Zhang, "Generic Multicast Router Election on LAN's", draft-wijnands-bier-mld-lan-election-01 (work in progress), July 2016.
- [RFC6826] Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M. Napierala, "Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013, <<http://www.rfc-editor.org/info/rfc6826>>.
- [RFC7431] Karan, A., Filsfils, C., Wijnands, IJ., Ed., and B. Decraene, "Multicast-Only Fast Reroute", RFC 7431, DOI 10.17487/RFC7431, August 2015, <<http://www.rfc-editor.org/info/rfc7431>>.
- [RFC7938] Lapukhov, P., Premji, A., and J. Mitchell, Ed., "Use of BGP for Routing in Large-Scale Data Centers", RFC 7938, DOI 10.17487/RFC7938, August 2016, <<http://www.rfc-editor.org/info/rfc7938>>.

Authors' Addresses

Zhaohui Zhang
Juniper Networks

EMail: zzhang@juniper.net

Keyur Patel
Arrcus

EMail: keyur@arrcus.com

IJsbrand Wijnands
Cisco Systems

EMail: ice@cisco.com

Arkadiy Gulko
Thomson Reuters

EMail: arkadiy.gulko@thomsonreuters.com