PIM WG Z. Zhang
Internet-Draft ZTE Corporation
Intended status: Standards Track F. Hu
Expires: April 3, 2021 Individual
B. Xu
ZTE Corporation
M. Mishra
Cisco Systems
September 30, 2020

Protocol Independent Multicast - Sparse Mode (PIM-SM) Designated Router (DR) Improvement

draft-ietf-pim-dr-improvement-10

Abstract

Protocol Independent Multicast - Sparse Mode (PIM-SM) is a widely deployed multicast protocol. As deployment for the PIM protocol is growing day by day, a user expects lower packet loss and faster convergence regardless of the cause of the network failure. This document defines an extension to the existing protocol, which improves the PIM protocol's stability with respect to packet loss and convergence time when the PIM Designated Router (DR) role changes.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Multicast technology, with PIM-SM ([RFC7761]), is used widely in modern services like IPTV and Net-Meeting. Some events, such as changes in unicast routes, or a change in the PIM-SM DR, may cause the loss of multicast packets.

The PIM DR has two responsibilities in the PIM-SM protocol. For any active sources on a LAN, the PIM DR is responsible for registering with the Rendezvous Point (RP) if the group is operating in PIM-SM. Also, the PIM DR is responsible for tracking local multicast listeners and forwarding data to these listeners if the group is operating in PIM-SM.

```
                           +             +
                           |             |
                     +-----+----+  +-----+----+
                     | RouterA  |  | RouterB  |
                     +-----+----+  +-----+----+
                           |             |
                 +----+----+--------+---------+---+----+
                      |             |             |
                      +             +             +
                  Receiver1     Receiver2     Receiver3
                 Figure 1: An example of multicast network
```

The simple network in Figure 1 presents two routers (A and B) connected to a shared-media LAN segment. Two different scenarios are described to illustrate potential issues.

(a) Both routers are on the network, and RouterB is elected as the DR. If RouterB then fails, multicast packets are discarded until RouterA is elected as DR and it assumes the multicast flows on the LAN. As detailed in [RFC7761], a DR's election is triggered after the current DR's Hello_Holdtime expires. When the DR (RouterB) is deemed unavailable, as the result of DR failure detection, RouterA is elected as the DR. Then RouterA joins the multicast trees, starts receiving the flows and proceeds with the multicast forwarding. All the procedures usually take several seconds. That is too long for modern multicast services.

(b) Only RouterA is initially on the network, making it the DR. If RouterB joins the network with a higher DR Priority. Then it will then be elected as DR. RouterA will stop forwarding multicast packets, and the multicast flows will not recover until RouterB assumes the multicast flows on the LAN.

In either of the situations listed, many multicast packets are lost, and the quality of multicast services noticeably affected. To increase the stability of the network, this document introduces the Designated DR (DR) and Backup Designated Router (BDR) options and specifies how its identity is explicitly advertised.

1.1. Keywords

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

2. Terminology

Backup Designated Router (BDR): Immediately takes over all DR functions ([RFC7761]) on an interface once the DR is no longer present. A single BDR SHOULD be elected per interface.

Designated Router Other (DROther): A router which is neither a DR nor a BDR.

0x0: 0.0.0.0 if IPv4 addresses are in use or 0:0:0:0:0:0:0:0/128 if IPv6 addresses are in use. To simplify, 0x0 is used in abbreviation in this draft.

3. Protocol Specification

The router follows the following procedures:

(a). A router first starts sending Hello messages with the values of DR and BDR Address options are all set to 0x0, after its interface is enabled in PIM on a shared-media LAN. The router treats itself as DROther role, and starts a timer which value is set to Hello_Holdtime.

(b). When the router receives Hello messages from other routers on the same shared-media LAN, the router checks the value of DR/BDR Address option. If the value is filled with a non-zero IP address, the router stores the IP addresses.

(c). When a Hello message with a non-zero DR Address option is received or after the timer expires, the router first executes the algorithm defined in section 3.1. After that, the router first one of the roles in the LAN: DR, BDR, or DROther.

If the role of the router first starts changes to BDR, the following steps are:

- o The BDR takes on all the functions of a DR as specified in [RFC7761], except that it SHOULD NOT actively forward multicast flows or send a register message to avoid duplication.

- o If the DR becomes unreachable on the LAN, the BDR MUST take over all the DR functions, including multicast flow forwarding or send the register message. Mechanisms outside the scope of this specification, such as [I-D.ietf-pim-bfd-p2mp-use-case] or BFD Asynchronous mode [RFC5880] can be used for faster failure detection.

For example, there are three routers: A, B, and C. If all three were in the LAN, then their DR preference would be A, B, and C, in that order. Initially, only C is on the LAN, so C is DR. Later, A joins; C is still the DR, and A is the BDR. Later B joins, and if B is better than A, then B becomes the BDR, and A is simply DROther.

3.1. Election Algorithm

The DR and BDR election is according to the DR election algorithm defined in section 9.4 in [RFC2328], except:

- o The DR is elected among the DR candidates directly. If there is no DR candidates, i.e., no router advertise the DR Address options with a non-zero IP address, the elected BDR will be the DR. And then the BDR is elected again from the other routers in the LAN.

- o The BDR election is not sticky. Whatever there is a router that advertise the BDR Address option, the router which has the highest priority, expect the elected DR, is elected as the BDR. That is the BDR may be the router which has the highest priority in the LAN.

- o The advertisement is through PIM Hello message.

- o Step 6 and 7 in section 9.4 in [RFC2328] are not applicable here.

Compare to the DR election function defined in section 4.3.2 in [RFC7761] the differences include:

- o The router, that can be elected as DR, has the highest priority among the DR candidates. The elected DR may not be the one that has the highest priority in the LAN.

- o The router that supports the election algorithm defined in section 3.1 MUST advertise the DR Address option defined in section 4.1 in PIM Hello message, and SHOULD advertise the BDR Address option defined in section 4.2 in PIM Hello message. In case a DR is elected and no BDR is elected, only the DR Address option is advertised in the LAN.

3.2. Sending Hello Messages

When PIM is enabled on an interface or a router first starts, Hello messages MUST be sent with the values of the DR Address option filled with 0x0. The BDR Address option SHOULD be sent, if the option is carried, the value MUST be filled with 0x0. Then the interface starts a timer which value is set to Hello_Holdtime. When the timer expires, the DR and BDR will be elected on the interface according to the DR election algorithm (Section 3.1).

```
                  DR                                newcomer
                   \                                   /
                    -----       -----           -----
                    | A |       | B |           | C |
                    -----       -----           -----
                      |           |               |
                      |           |               |
              ------------------------------------------- LAN
                                Figure 2
```

For example, there is a stable LAN that includes RouterA and RouterB. RouterA is the DR that has the highest priority. RouterC is a newcomer. RouterC sends a Hello message with the DR and BDR Address options are all set to zero.

When a router first starts (RouterC) elects itself as the BDR after it running the election algorithm, the router sends Hello messages with the value of DR is set to the IP address of current DR (RouterA) and the value of BDR is set to the IP address of the router first starts itself (RouterC).

A current BDR (RouterB) may find that it can not be the BDR after it running the election algorithm, it MUST set itself DROther and stop sending the BDR Address options with its IP address. It MUST send Hello messages with the value of DR is set to current DR and the value of BDR is set to the newly elected BDR.

3.3. Receiving Hello Messages

When a Hello message is received, if the DR/BDR Address option carried in the message is different from the previous message. The election algorithm MUST be rerun. As a result, the associate actions should be taken according to the role changing.

3.4. Working with the DRLB function

The DRLB function defined in [RFC8775] can work with the mechanism defined in this document. The routers advertise the DR/BDR Address options and the DRLB-Cap Hello Option defined in [RFC8775]. After running the election algorithm defined in section 3.1, the elected DR advertises the DRLB-List Hello Option to carry the GDR candidates.

When the current DR is unavailable, the BDR MUST send the DRLB-List Hello Option to carry the GDR candidates. The BDR starts forwarding the multicast flows, but there may be duplicated flows because the DR may not be the same as the GDR.

4. PIM Hello message format

Two new PIM Hello Options are defined, which conform to the format defined in [RFC7761].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          OptionType           |         OptionLength          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          OptionValue                          |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: Hello Option Format

4.1. DR Address Option format

- o OptionType : The value is 37.
- o OptionLength: 4 bytes if using IPv4 and 16 bytes if using IPv6.
- o OptionValue: IP address of the DR. If the IP version of the PIM message is IPv4, the value MUST be the IPv4 address of the DR. If the IP version of the PIM message is IPv6, the value MUST be the link-local address of the DR.

4.2. BDR Address Option format

- o OptionType : The value is 38.
- o OptionLength: 4 bytes if using IPv4 and 16 bytes if using IPv6.
- o OptionValue: IP address of the BDR. If the IP version of the PIM message is IPv4, the value MUST be the IPv4 address of the BDR. If the IP version of the PIM message is IPv6, the value MUST be the link-local address of the BDR.

4.3. Error handling

The DR and BDR addresses MUST be the same with the addresses which are used to send PIM Hello message.

Unknown options MUST be ignored, which conforms to the format defined in section 4.9.2 in [RFC7761], and the options MUST be ignored that include unexpected values. For example, when a DR Address option with IPv4 address is received while the interface supports IPv6 only, the option MUST be ignored.

5. Compatibility

If at least one router on a LAN doesn't send a Hello message, including the DR Address Options, then the specification in this document MUST NOT be used. Any router using the DR and BDR Address Options MUST set the corresponding OptionValues to 0x0. This action results in all routers using the DR election function defined in [RFC7761] or [I-D.mankamana-pim-bdr].

This draft allows the DR election to be sticky by not unnecessarily changing the DR when routers go down or come up. That is done by introducing new PIM Hello options. Both this draft and the draft [I-D.mankamana-pim-bdr], introduce a backup DR. The latter draft does this without introducing new options but does not consider the sticky behavior.

A router that does not support this specification ignores unknown options According to section 4.9.2 defined in [RFC7761]. So the new extension defined in this draft will not influence the stability of neighbors.

The DR election mechanism selection would depend on deployment scenario.

6. Security Considerations

[RFC7761] describes the security concerns related to PIM-SM, the potential BFD session attack can be used as the security function in section 9 [RFC5880] mentioned.

If an attacker wants to hijack the DR role, it may send PIM Hello message with the altered DR/BDR Address options. The attacker sends the Hello message with the DR Address option set to itself as DR except for the highest priority or IP address. Or the attacker sends the Hello message without the DR/BDR Address option except for the highest priority or IP address.

If an attacker wants to take the BDR role, it simply sends PIM Hello message with BDR Address options except for the higher priority or IP address than the current BDR.

Some security measures, such as IP address filtering for the election, may be taken to avoid these situations. For example, the Hello message received from an unknown neighbor is ignored by the election process.

7. IANA Considerations

IANA is requested to allocate two new code points from the "PIM-Hello Options" registry.

Type	Description	Reference
37	DR Address Option	This Document
38	BDR Address Option	This Document

Table 1

8. Acknowledgements

The authors would like to thank Alvaro Retana, Greg Mirsky, Jake Holland, Stig Venaas for their valuable comments and suggestions.

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>.

[RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <https://www.rfc-editor.org/info/rfc2328>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[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>.

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

[RFC8775] Cai, Y., Ou, H., Vallepalli, S., Mishra, M., Venaas, S., and A. Green, "PIM Designated Router Load Balancing", RFC 8775, DOI 10.17487/RFC8775, April 2020, <https://www.rfc-editor.org/info/rfc8775>.

9.2. Informative References

[I-D.ietf-pim-bfd-p2mp-use-case]
Mirsky, G. and J. Xiaoli, "Bidirectional Forwarding Detection (BFD) for Multi-point Networks and Protocol Independent Multicast - Sparse Mode (PIM-SM) Use Case", draft-ietf-pim-bfd-p2mp-use-case-04 (work in progress), July 2020.

[I-D.mankamana-pim-bdr]
mishra, m., Goh, J., and G. Mishra, "PIM Backup Designated Router Procedure", draft-mankamana-pim-bdr-04 (work in progress), April 2020.

Authors' Addresses

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

Email: zhang.zheng@zte.com.cn

Fangwei Hu
Individual
Shanghai
China

Email: hufwei@gmail.com

Benchong Xu
ZTE Corporation
No. 68 Zijinghua Road, Yuhuatai Distinct
Nanjing
China

Email: xu.benchong@zte.com.cn

Mankamana Mishra
Cisco Systems
821 Alder Drive,
MILPITAS, CALIFORNIA 95035
UNITED STATES

Email: mankamis@cisco.com

PIM Working Group
Internet Draft
Intended status: Standards Track
Expires: February 14, 2021

H. Zhao
Ericsson
X. Liu
Volta Networks
Y. Liu
China Mobile
M. Sivakumar
Juniper
A. Peter
Individual

August 15, 2020

A Yang Data Model for IGMP and MLD Snooping
draft-ietf-pim-igmp-mld-snooping-yang-18.txt

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) Snooping devices. The YANG module in this document conforms to Network Management Datastore Architecture (NMDA).

Status of this Memo

Table of Contents

1. Introduction

This document defines a YANG [RFC7950] data model for the management of Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping [RFC4541] devices.

The YANG module in this document conforms to the Network Management Datastore Architecture defined in [RFC8342]. The "Network Management Datastore Architecture" (NMDA) adds the ability to inspect the current operational values for configuration, allowing clients to use identical paths for retrieving the configured values and the operational values.

1.1. Terminology

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

and [RFC7950], including:

* augment

* data model

* data node

* identity

* module

The following terminologies are used in this document:

* mrouter: multicast router, which is a router that has multicast routing enabled [RFC4286].

* mrouter interfaces: snooping switch ports where multicast routers are attached [RFC4541].

The following abbreviations are used in this document and defined model:

IGMP: Internet Group Management Protocol [RFC3376].

MLD: Multicast Listener Discovery [RFC3810].

AC: Attachment Circuit [RFC3916].

PW: Pseudo Wire [RFC3916].

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in

[RFC8340].

1.3. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
inet	ietf-inet-types	[RFC6991]
yang	ietf-yang-types	[RFC6991]
if	ietf-interfaces	[RFC8343]
rt	ietf-routing	[RFC8349]
rt-types	ietf-routing-types	[RFC8294]
ni	ietf-network-instance	[RFC8529]
pw	ietf-pseudowires	[draft-ietf-bess-l2vpn-yang]
l2vpn	ietf-l2vpn	[draft-ietf-bess-l2vpn-yang]
dot1q	ieee802-dot1q-bridge	[dot1Qcp]

Table 1: Prefixes and Corresponding YANG Modules

2. Design of Data Model

An IGMP/MLD snooping switch [RFC4541] analyzes IGMP/MLD packets and sets up forwarding tables for multicast traffic. If a switch does not run IGMP/MLD snooping, multicast traffic will be flooded in the broadcast domain. If a switch runs IGMP/MLD snooping, multicast traffic will be forwarded based on the forwarding tables to avoid wasting bandwidth. The IGMP/MLD snooping switch does not need to run any of the IGMP/MLD

protocols. Because the IGMP/MLD snooping is independent of the IGMP/MLD protocols, the data model defined in this document does not augment, or even require, the IGMP/MLD data model defined in [RFC8652].
The model covers considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches [RFC4541].

IGMP and MLD snooping switches do not adhere to the conceptual model that provides the strict separation of functionality between different communications layers in the ISO model, and instead utilize information in the upper level protocol headers as factors to be considered in processing at the lower levels [RFC4541].

IGMP Snooping switches utilize IGMP, and could support IGMPv1 [RFC1112], IGMPv2 [RFC2236], and IGMPv3 [RFC3376]. MLD Snooping switches utilize MLD, and could support MLDv1 [RFC2710] and MLDv2 [RFC3810]. The goal of this document is to define a data model that provides a common user interface to IGMP and MLD Snooping.

2.1. Overview

The IGMP and MLD Snooping YANG module defined in this document has all the common building blocks for the IGMP and MLD Snooping switches.

The YANG module includes IGMP and MLD Snooping instance definition, using instance in the L2 service type of BRIDGE [dot1Qcp] and L2VPN [draft-ietf-bess-l2vpn-yang]. The module also includes actions for clearing IGMP and MLD Snooping group tables.

2.2. Optional Capabilities

This model is designed to represent the basic capability subsets of IGMP and MLD Snooping. The main design goals of this document are that the basic capabilities described in the model are supported by any major now-existing implementation, and that the configuration of all implementations meeting the specifications is easy to express through some combination of the optional features in the model and simple vendor augmentations.

There is also value in widely supported features being standardized, to provide a standardized way to access these features, to save work for individual vendors, and so that mapping between different vendors' configuration is not needlessly complicated. Therefore, this model declares 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 Snooping 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.

2.3. Position of Address Family in Hierarchy

IGMP Snooping only supports IPv4, while MLD Snooping only supports IPv6. The data model defined in this document can be used for both IPv4 and IPv6 address families.

This document defines IGMP Snooping and MLD Snooping as separate schema branches in the structure. The benefits are:

* The model can support IGMP Snooping (IPv4), MLD Snooping (IPv6), or both optionally and independently. Such flexibility cannot be achieved cleanly with a combined branch.

* The structure is consistent with other YANG data models such as [RFC8652], which uses separate branches for IPv4 and IPv6.

* Having separate branches for IGMP Snooping and MLD Snooping allows minor differences in their behavior to be modelled more simply and cleanly. The two branches can better support different features and node types.

3. Module Structure

This model augments the core routing data model specified in [RFC8349].

```
      +--rw routing
         +--rw router-id?
         +--rw control-plane-protocols
         |  +--rw control-plane-protocol* [type name]
         |     +--rw type
         |     +--rw name
         |     +--rw igmp-snooping-instance <= Augmented by this Model
                        ...
         |     +--rw mld-snooping-instance   <= Augmented by this Model
                        ...
```

The "igmp-snooping-instance" container instantiates an IGMP Snooping Instance. The "mld-snooping-instance" container instantiates an MLD Snooping Instance.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407].

3.1. IGMP Snooping Instances

The YANG module ietf-igmp-mld-snooping augments /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol to add the igmp-snooping-instance container.

All the IGMP Snooping related attributes have been defined in the igmp-snooping-instance. The read-write attributes represent configurable data. The read-only attributes represent state data.

One igmp-snooping-instance could be used in one BRIDGE [dot1Qcp] instance or L2VPN [draft-ietf-bess-l2vpn-yang] instance. One igmp-snooping-instance corresponds to one BRIDGE instance or one L2VPN instance.

The value of l2-service-type in igmp-snooping-instance is bridge or l2vpn. When it is bridge, igmp-snooping-instance will be used in the BRIDGE service. When it is l2vpn, igmp-snooping-instance will be used in the L2VPN service.

The values of bridge-mrouter-interface, l2vpn-mrouter-interface-ac, l2vpn-mrouter-interface-pw are filled by the snooping device dynamically. They are different from static-bridge-mrouter-interface, static-l2vpn-mrouter-interface-ac, and static-l2vpn-mrouter-interface-pw which are configured.

The attributes under the interfaces show the statistics of IGMP Snooping related packets.

```
  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw igmp-snooping-instance {igmp-snooping}?
       +--rw l2-service-type?                    l2-service-type
       +--rw enable?                             boolean
       +--rw forwarding-table-type?              enumeration
       +--rw explicit-tracking?                  boolean
       |       {explicit-tracking}?
       +--rw lite-exclude-filter?                empty
       |       {lite-exclude-filter}?
       +--rw send-query?                         boolean
       +--rw immediate-leave?                    empty
       |       {immediate-leave}?
       +--rw last-member-query-interval?         uint16
       +--rw query-interval?                     uint16
       +--rw query-max-response-time?            uint16
       +--rw require-router-alert?               boolean
       |       {require-router-alert}?
       +--rw robustness-variable?                uint8
       +--rw static-bridge-mrouter-interface*    if:interface-ref
       |       {static-mrouter-interface}?
       +--rw static-l2vpn-mrouter-interface-ac*   if:interface-ref
       |       {static-mrouter-interface}?
```

```
       +--rw static-l2vpn-mrouter-interface-pw*   pw:pseudowire-ref
       |       {static-mrouter-interface}?
       +--rw igmp-version?                        uint8
       +--rw querier-source?                      inet:ipv4-address
       +--rw static-l2-multicast-group* [group source-addr]
       |       {static-l2-multicast-group}?
       |  +--rw group
       |  |       rt-types:ipv4-multicast-group-address
       |  +--rw source-addr
       |  |       rt-types:ipv4-multicast-source-address
       |  +--rw bridge-outgoing-interface*   if:interface-ref
       |  +--rw l2vpn-outgoing-ac*           if:interface-ref
       |  +--rw l2vpn-outgoing-pw*           pw:pseudowire-ref
       +--ro entries-count?                       yang:gauge32
       +--ro bridge-mrouter-interface*            if:interface-ref
       +--ro l2vpn-mrouter-interface-ac*          if:interface-ref
       +--ro l2vpn-mrouter-interface-pw*          pw:pseudowire-ref
       +--ro group* [address]
       |  +--ro address
       |  |       rt-types:ipv4-multicast-group-address
       |  +--ro mac-address?     yang:phys-address
       |  +--ro expire?          rt-types:timer-value-seconds16
       |  +--ro up-time          uint32
       |  +--ro last-reporter?   inet:ipv4-address
       |  +--ro source* [address]
       |     +--ro address
       |     |       rt-types:ipv4-multicast-source-address
       |     +--ro bridge-outgoing-interface*   if:interface-ref
       |     +--ro l2vpn-outgoing-ac*           if:interface-ref
       |     +--ro l2vpn-outgoing-pw*           pw:pseudowire-ref
       |     +--ro up-time                      uint32
       |     +--ro expire?
       |     |       rt-types:timer-value-seconds16
       |     +--ro host-count?                  yang:gauge32
       |     |       {explicit-tracking}?
       |     +--ro last-reporter?               inet:ipv4-address
       |     +--ro host* [host-address] {explicit-tracking}?
       |        +--ro host-address        inet:ipv4-address
       |        +--ro host-filter-mode    filter-mode-type
       +--ro interfaces
          +--ro interface* [name]
             +--ro name          if:interface-ref
             +--ro statistics
                +--ro discontinuity-time?   yang:date-and-time
                +--ro received
                |  +--ro query-count?                    yang:counter64
                |  +--ro membership-report-v1-count?   yang:counter64
                |  +--ro membership-report-v2-count?   yang:counter64
                |  +--ro membership-report-v3-count?   yang:counter64
                |  +--ro leave-count?                    yang:counter64
                |  +--ro pim-hello-count?                yang:counter64
                +--ro sent
```

```
                   +--ro query-count?                    yang:counter64
                   +--ro membership-report-v1-count?   yang:counter64
                   +--ro membership-report-v2-count?   yang:counter64
                   +--ro membership-report-v3-count?   yang:counter64
                   +--ro leave-count?                    yang:counter64
                   +--ro pim-hello-count?                yang:counter64
```

3.2. MLD Snooping Instances

The YANG module ietf-igmp-mld-snooping augments /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol to add the mld-snooping-instance container. The mld-snooping-instance could be used in the BRIDGE [dot1Qcp] or L2VPN [draft-ietf-bess-l2vpn-yang] service to enable MLD Snooping.

All the MLD Snooping related attributes have been defined in the mld-snooping-instance. The read-write attributes represent configurable data. The read-only attributes represent state data.

The mld-snooping-instance has similar structure as IGMP snooping. Some of leaves are protocol related. The mld-snooping-instance uses IPv6 addresses and mld-version, while igmp-snooping-instance uses IPv4 addresses and igmp-version. Statistic counters in each of the above snooping instances are also tailored to the specific protocol type. One mld-snooping-instance could be used in one BRIDGE instance or L2VPN instance. One mld-snooping-instance corresponds to one BRIDGE instance or L2VPN instance.

The value of l2-service-type in mld-snooping-instance is bridge or l2vpn. When it is bridge, mld-snooping-instance will be used in the BRIDGE service. When it is l2vpn, mld-snooping-instance will be used in the L2VPN service.

The values of bridge-mrouter-interface, l2vpn-mrouter-interface-ac, l2vpn-mrouter-interface-pw are filled by the snooping device dynamically. They are different from static-bridge-mrouter-interface, static-l2vpn-mrouter-interface-ac, and static-l2vpn-mrouter-interface-pw which are configured.

The attributes under the interfaces show the statistics of MLD Snooping related packets.

```
augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol:
  +--rw mld-snooping-instance {mld-snooping}?
       +--rw l2-service-type?                     l2-service-type
       +--rw enable?                              boolean
       +--rw forwarding-table-type?               enumeration
       +--rw explicit-tracking?                   boolean
       |       {explicit-tracking}?
       +--rw lite-exclude-filter?                 empty
```

```
       |       {lite-exclude-filter}?
       +--rw send-query?                          boolean
       +--rw immediate-leave?                     empty
       |       {immediate-leave}?
       +--rw last-member-query-interval?          uint16
       +--rw query-interval?                      uint16
       +--rw query-max-response-time?             uint16
       +--rw require-router-alert?                boolean
       |       {require-router-alert}?
       +--rw robustness-variable?                 uint8
       +--rw static-bridge-mrouter-interface*     if:interface-ref
       |       {static-mrouter-interface}?
       +--rw static-l2vpn-mrouter-interface-ac*   if:interface-ref
       |       {static-mrouter-interface}?
       +--rw static-l2vpn-mrouter-interface-pw*   pw:pseudowire-ref
       |       {static-mrouter-interface}?
       +--rw mld-version?                         uint8
       +--rw querier-source?                      inet:ipv6-address
       +--rw static-l2-multicast-group* [group source-addr]
       |       {static-l2-multicast-group}?
       |  +--rw group
       |  |       rt-types:ipv6-multicast-group-address
       |  +--rw source-addr
       |  |       rt-types:ipv6-multicast-source-address
       |  +--rw bridge-outgoing-interface*   if:interface-ref
       |  +--rw l2vpn-outgoing-ac*           if:interface-ref
       |  +--rw l2vpn-outgoing-pw*           pw:pseudowire-ref
       +--ro entries-count?                       yang:gauge32
       +--ro bridge-mrouter-interface*            if:interface-ref
       +--ro l2vpn-mrouter-interface-ac*          if:interface-ref
       +--ro l2vpn-mrouter-interface-pw*          pw:pseudowire-ref
       +--ro group* [address]
       |  +--ro address
       |  |       rt-types:ipv6-multicast-group-address
       |  +--ro mac-address?     yang:phys-address
       |  +--ro expire?          rt-types:timer-value-seconds16
       |  +--ro up-time          uint32
       |  +--ro last-reporter?   inet:ipv6-address
       |  +--ro source* [address]
       |     +--ro address
       |     |       rt-types:ipv6-multicast-source-address
       |     +--ro bridge-outgoing-interface*   if:interface-ref
       |     +--ro l2vpn-outgoing-ac*           if:interface-ref
       |     +--ro l2vpn-outgoing-pw*           pw:pseudowire-ref
       |     +--ro up-time                      uint32
       |     +--ro expire?
       |     |       rt-types:timer-value-seconds16
       |     +--ro host-count?                  yang:gauge32
       |     |       {explicit-tracking}?
       |     +--ro last-reporter?               inet:ipv6-address
       |     +--ro host* [host-address] {explicit-tracking}?
       |        +--ro host-address        inet:ipv6-address
```

```
       |        +--ro host-filter-mode    filter-mode-type
       +--ro interfaces
          +--ro interface* [name]
             +--ro name          if:interface-ref
             +--ro statistics
                +--ro discontinuity-time?   yang:date-and-time
                +--ro received
                |  +--ro query-count?       yang:counter64
                |  +--ro report-v1-count?   yang:counter64
                |  +--ro report-v2-count?   yang:counter64
                |  +--ro done-count?        yang:counter64
                |  +--ro pim-hello-count?   yang:counter64
                +--ro sent
                   +--ro query-count?       yang:counter64
                   +--ro report-v1-count?   yang:counter64
                   +--ro report-v2-count?   yang:counter64
                   +--ro done-count?        yang:counter64
                   +--ro pim-hello-count?   yang:counter64
```

3.3. Using IGMP and MLD Snooping Instances

The igmp-snooping-instance could be used in the service of BRIDGE [dot1Qcp] or L2VPN [draft-ietf-bess-l2vpn-yang] to configure the IGMP Snooping.

For the BRIDGE service this model augments /dot1q:bridges/dot1q:bridge to use igmp-snooping-instance. It means IGMP Snooping is enabled in the whole bridge.

It also augments /dot1q:bridges/dot1q:bridge/dot1q:component/ dot1q:bridge-vlan/dot1q:vlan to use igmp-snooping-instance. It means IGMP Snooping is enabled in the specified VLAN on the bridge.

```
     augment /dot1q:bridges/dot1q:bridge:
       +--rw igmp-snooping-instance?   igmp-mld-snooping-instance-ref
       +--rw mld-snooping-instance?    igmp-mld-snooping-instance-ref

     augment /dot1q:bridges/dot1q:bridge/dot1q:component
               /dot1q:bridge-vlan/dot1q:vlan:
       +--rw igmp-snooping-instance?   igmp-mld-snooping-instance-ref
       +--rw mld-snooping-instance?    igmp-mld-snooping-instance-ref
```

For the L2VPN service this model augments /ni:network-instances/ ni:network-instance/ni:ni-type/l2vpn:l2vpn [RFC8529] to use igmp-snooping-instance. It means IGMP Snooping is enabled in the specified l2vpn instance.

```
     augment /ni:network-instances/ni:network-instance/ni:ni-type
               /l2vpn:l2vpn:
       +--rw igmp-snooping-instance?   igmp-mld-snooping-instance-ref
       +--rw mld-snooping-instance?    igmp-mld-snooping-instance-ref
```

The mld-snooping-instance could be used in concurrence with igmp-snooping-instance to configure the MLD Snooping.

3.4. IGMP and MLD Snooping Actions

IGMP and MLD Snooping actions clear the specified IGMP and MLD Snooping group tables. If both source X and group Y are specified, only source X from group Y in that specific instance will be cleared.

```
augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol:
    +--rw igmp-snooping-instance {igmp-snooping}?
      +---x clear-igmp-snooping-groups {action-clear-groups}?
          +---w input
             +---w group     union
             +---w source    rt-types:ipv4-multicast-source-address

augment /rt:routing/rt:control-plane-protocols
           /rt:control-plane-protocol:
    +--rw mld-snooping-instance {mld-snooping}?
       +---x clear-mld-snooping-groups {action-clear-groups}?
          +---w input
             +---w group     union
             +---w source    rt-types:ipv6-multicast-source-address
```

4. IGMP and MLD Snooping YANG Module

This module references [RFC1112],[RFC2236],[RFC2710],[RFC3376],[RFC3810],[RFC4541],[RFC5790],[RFC6636],[RFC6991],[RFC7761],[RFC8343],[RFC8529],[dot1Qcp], and [draft-ietf-bess-l2vpn-yang].

```
<CODE BEGINS> file ietf-igmp-mld-snooping@2020-08-07.yang
module ietf-igmp-mld-snooping {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld-snooping";

  prefix ims;

  import ietf-inet-types {
    prefix "inet";
    reference
      "RFC 6991: Common YANG Data Types";
  }
```

```
  import ietf-yang-types {
    prefix "yang";
    reference
      "RFC 6991: Common YANG Data Types";
  }

  import ietf-interfaces {
    prefix "if";
    reference
      "RFC 8343: A YANG Data Model for Interface Management";
  }

  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management (NMDA
       Version)";
  }

  import ietf-routing-types {
    prefix "rt-types";
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
    reference
      "draft-ietf-bess-l2vpn-yang: YANG Data Model for MPLS-based
L2VPN";
  }

  import ietf-network-instance {
    prefix "ni";
    reference
      "RFC 8529: YANG Data Model for Network Instances";
  }

  import ietf-pseudowires {
    prefix "pw";
    reference
      "draft-ietf-bess-l2vpn-yang: YANG Data Model for MPLS-based
L2VPN";
  }

  import ieee802-dot1q-bridge {
    prefix "dot1q";
    reference
      "dot1Qcp: IEEE 802.1Qcp-2018 Bridges and Bridged Networks
                - Amendment: YANG Data Model";
  }
```

```
  organization
    "IETF PIM Working Group";

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

     Editors:  Hongji Zhao
               <mailto:hongji.zhao@ericsson.com>

               Xufeng Liu
               <mailto:xufeng.liu.ietf@gmail.com>

               Yisong Liu
               <mailto:liuyisong@chinamobile.com>

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

               Mahesh Sivakumar
               <mailto:sivakumar.mahesh@gmail.com>

    ";

  description
    "The module defines a collection of YANG definitions common for
     all devices that implement Internet Group Management Protocol
     (IGMP) and Multicast Listener Discovery (MLD) Snooping which is
     described in RFC 4541.

     Copyright (c) 2020 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX; see the
     RFC itself for full legal notices.";

  revision 2020-08-07 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";
  }
```

```
  /*
   * Features
   */

  feature igmp-snooping {
    description
      "Support IGMP snooping.";
    reference
      "RFC 4541";
  }

  feature mld-snooping {
    description
      "Support MLD snooping.";
    reference
      "RFC 4541";
  }

  feature immediate-leave {
    description
      "Support configuration of fast leave. The fast leave feature
       does not send last member query messages to hosts.";
    reference
      "RFC 3376";
  }

  feature static-l2-multicast-group {
    description
      "Support configuration of L2 multicast static-group.";
  }

  feature static-mrouter-interface {
    description
      "Support multicast router interface explicitly configured
       by management";
    reference
      "RFC 4541";
  }

  feature action-clear-groups {
    description
      "Support clearing statistics by action for IGMP & MLD snooping.";
  }

  feature require-router-alert {
    description
      "Support configuration of require-router-alert.";
    reference
      "RFC 3376";
  }

  feature lite-exclude-filter {
```

```
    description
      "Enable the support of the simplified EXCLUDE filter.";
    reference
      "RFC 5790";
  }

  feature explicit-tracking {
    description
      "Support configuration of per instance explicit-tracking.";
    reference
      "RFC 6636";
  }

  /* identities */

  identity l2-service-type {
    description
      "Base identity for L2 service type in IGMP & MLD snooping";
  }

  identity bridge {
    base l2-service-type;
    description
      "This identity represents BRIDGE service.";
  }

  identity l2vpn {
    base l2-service-type;
    description
      "This identity represents L2VPN service.";
  }

  identity filter-mode {
    description
      "Base identity for filter mode in IGMP & MLD snooping";
  }

  identity include {
    base filter-mode;
    description
      "This identity represents include mode.";
  }

  identity exclude {
    base filter-mode;
    description
      "This identity represents exclude mode.";
  }

  identity igmp-snooping {
    base rt:control-plane-protocol;
    description
```

```
      "IGMP snooping";
  }

  identity mld-snooping {
    base rt:control-plane-protocol;
    description
      "MLD snooping";
  }

  /*
   * Typedefs
   */

  typedef l2-service-type {
    type identityref {
      base "l2-service-type";
    }
    description "The L2 service type used with IGMP & MLD snooping ";
  }

  typedef filter-mode-type {
    type identityref {
      base "filter-mode";
    }
    description "The host filter mode";
  }

  typedef igmp-mld-snooping-instance-ref {
    type leafref {
      path "/rt:routing/rt:control-plane-protocols"+
        "/rt:control-plane-protocol/rt:name";
    }
    description
      "This type is used by data models which need to
       reference IGMP & MLD snooping instance.";
  }

  /*
   * Groupings
   */


  grouping instance-config-attributes-igmp-mld-snooping {
    description
      "IGMP and MLD snooping configuration of each VLAN.";

    leaf enable {
      type boolean;
      default false;
        description
          "Set the value to true to enable IGMP & MLD snooping.";
    }
```

```
    leaf forwarding-table-type {
      type enumeration {
        enum "mac" {
          description
          "MAC-based lookup mode";
        }
        enum "ip"  {
          description
          "IP-based lookup mode";
        }
      }
      default "ip";
      description "The default forwarding table type is ip";
    }

    leaf explicit-tracking {
      if-feature explicit-tracking;
      type boolean;
      default false;
      description
        "Track the IGMPv3 and MLDv2 snooping membership reports
         from individual hosts. It contributes to saving network
         resources and shortening leave latency.";
    }

    leaf lite-exclude-filter {
      if-feature lite-exclude-filter;
      type empty;
      description
        "For IGMP Snooping, the presence of this
         leaf enables the support of the simplified EXCLUDE filter
         in the Lightweight IGMPv3 protocol, which simplifies the
         standard versions of IGMPv3.
         For MLD Snooping, the presence of this
         leaf enables the support of the simplified EXCLUDE filter
         in the Lightweight MLDv2 protocol, which simplifies the
         standard versions of MLDv2.";
      reference
        "RFC 5790";
    }

    leaf send-query {
      type boolean;
      default false;
      description
        "Enable quick response for topology changes.
         To support IGMP snooping in a VLAN where PIM and IGMP are
         not configured. It cooperates with parameter querier-source.";
    }

    leaf immediate-leave {
```

```
      if-feature immediate-leave;
      type empty;
      description
        "When immediate leave is enabled, the IGMP software assumes
         that no more than one host is present on each VLAN port.";
    }

    leaf last-member-query-interval {
      type uint16 {
        range "10..10230";
      }
      units deciseconds;
      default 10;
      description
        "Last Member Query Interval, which may be tuned to modify
         the leave latency of the network.
         It is represented in units of 1/10 second.";
      reference "RFC 3376. Sec. 8.8.";
    }

    leaf query-interval {
      type uint16;
      units seconds;
      default 125;
      description
        "The Query Interval is the interval between General Queries
         sent by the Querier.";
      reference "RFC 3376. Sec. 4.1.7, 8.2, 8.14.2.";
    }

    leaf query-max-response-time {
      type uint16;
      units deciseconds;
      default 100;
      description
        "Query maximum response time specifies the maximum time
         allowed before sending a responding report.
         It is represented in units of 1/10 second.";
      reference "RFC 3376. Sec. 4.1.1, 8.3, 8.14.3.";
    }

    leaf require-router-alert {
      if-feature require-router-alert;
      type boolean;
      default false;
      description
        "When the value is true, router alert should exist
         in the IP header of IGMP or MLD packet.";
    }

    leaf robustness-variable {
      type uint8 {
```

```
        range "1..7";
      }
      default 2;
      description
        "Querier's Robustness Variable allows tuning for the
         expected packet loss on a network.";
      reference "RFC 3376. Sec. 4.1.6, 8.1, 8.14.1.";
    }

    leaf-list static-bridge-mrouter-interface {
      when 'derived-from-or-self(../l2-service-type,"ims:bridge")';
      if-feature static-mrouter-interface;
      type if:interface-ref;
      description "static mrouter interface in BRIDGE forwarding";
    }

    leaf-list static-l2vpn-mrouter-interface-ac {
      when 'derived-from-or-self(../l2-service-type,"ims:l2vpn")';
      if-feature static-mrouter-interface;
      type if:interface-ref;
      description
        "static mrouter interface whose type is interface
         in L2VPN forwarding";
    }

    leaf-list static-l2vpn-mrouter-interface-pw {
      when 'derived-from-or-self(../l2-service-type,"ims:l2vpn")';
      if-feature static-mrouter-interface;
      type pw:pseudowire-ref;
      description
        "static mrouter interface whose type is PW
         in L2VPN forwarding";
    }
  } // instance-config-attributes-igmp-mld-snooping

  grouping instance-state-group-attributes-igmp-mld-snooping {
    description
      "Attributes for both IGMP and MLD snooping groups.";

    leaf mac-address {
      type yang:phys-address;
      description "Destination MAC address for L2 multicast.";
    }

    leaf expire {
      type rt-types:timer-value-seconds16;
      units seconds;
      description
        "The time left before multicast group timeout.";
    }

    leaf up-time {
```

```
      type uint32;
      units seconds;
      mandatory true;
      description
        "The time elapsed since L2 multicast record created.";
    }
  } // instance-state-group-attributes-igmp-mld-snooping

  grouping instance-state-attributes-igmp-mld-snooping {

    description
      "State attributes for IGMP & MLD snooping instance.";

    leaf entries-count {
      type yang:gauge32;
      config false;
      description
        "The number of L2 multicast entries in IGMP & MLD snooping";
    }

    leaf-list bridge-mrouter-interface {
      when 'derived-from-or-self(../l2-service-type,"ims:bridge")';
      type if:interface-ref;
      config false;
      description
        "Indicates a list of mrouter interfaces dynamicly learned in a
         bridge. When this switch receives IGMP/MLD queries from a
         multicast router on an interface, the interface will become
         mrouter interface for IGMP/MLD snooping.";
    }

    leaf-list l2vpn-mrouter-interface-ac {
      when 'derived-from-or-self(../l2-service-type,"ims:l2vpn")';
      type if:interface-ref;
      config false;
      description
        "The mrouter interface whose type is interface in L2VPN
         forwarding. When switch receives IGMP/MLD queries from
         multicast router on an interface, this interface will
         become mrouter interface for IGMP/MLD snooping.";
    }

    leaf-list l2vpn-mrouter-interface-pw {
      when 'derived-from-or-self(../l2-service-type,"ims:l2vpn")';
      type pw:pseudowire-ref;
      config false;
      description
        "The mrouter interface whose type is PW in L2VPN forwarding.
         When switch receives IGMP/MLD queries from multicast router
         on a PW, this PW will become mrouter interface for IGMP/MLD
         snooping.";
    }
```

```
  } // instance-config-attributes-igmp-mld-snooping


  grouping instance-state-source-attributes-igmp-mld-snooping {
    description
      "State attributes for IGMP & MLD snooping instance.";

    leaf-list bridge-outgoing-interface {
      when 'derived-from-or-self(../../../l2-service-
type,"ims:bridge")';
      type if:interface-ref;
      description "Outgoing interface in BRIDGE forwarding";
    }

    leaf-list l2vpn-outgoing-ac {
      when 'derived-from-or-self(../../../l2-service-type,"ims:l2vpn")';
      type if:interface-ref;
      description "Outgoing Attachment Circuit (AC) in L2VPN";
    }

    leaf-list l2vpn-outgoing-pw {
      when 'derived-from-or-self(../../../l2-service-type,"ims:l2vpn")';
      type pw:pseudowire-ref;
      description "Outgoing Pseudo Wire (PW) in L2VPN";
    }

    leaf up-time {
      type uint32;
      units seconds;
      mandatory true;
      description
        "The time elapsed since L2 multicast record created";
    }

    leaf expire {
      type rt-types:timer-value-seconds16;
      units seconds;
      description
        "The time left before multicast group timeout.";
    }

    leaf host-count {
      if-feature explicit-tracking;
      type yang:gauge32;
      description
        "The number of host addresses.";
    }
  } // instance-state-source-attributes-igmp-mld-snooping

  grouping igmp-snooping-statistics {
    description
      "The statistics attributes for IGMP snooping.";
```

```
      leaf query-count {
        type yang:counter64;
        description
          "The number of Membership Query messages.";
        reference
          "RFC 2236";
      }
      leaf membership-report-v1-count {
        type yang:counter64;
        description
          "The number of Version 1 Membership Report messages.";
        reference
          "RFC 1112";
      }
      leaf membership-report-v2-count {
        type yang:counter64;
        description
          "The number of Version 2 Membership Report messages.";
        reference
          "RFC 2236";
      }
      leaf membership-report-v3-count {
        type yang:counter64;
        description
          "The number of Version 3 Membership Report messages.";
        reference
          "RFC 3376";
      }
      leaf leave-count {
        type yang:counter64;
        description
          "The number of Leave Group messages.";
        reference
          "RFC 2236";
      }
      leaf pim-hello-count {
        type yang:counter64;
        description
          "The number of PIM hello messages.";
        reference
          "RFC 7761";
      }
  } // igmp-snooping-statistics

  grouping mld-snooping-statistics {
    description
      "The statistics attributes for MLD snooping.";

      leaf query-count {
        type yang:counter64;
        description
```

```
          "The number of Multicast Listener Query messages.";
        reference
          "RFC 3810";
      }
      leaf report-v1-count {
        type yang:counter64;
        description
          "The number of Version 1 Multicast Listener Report.";
        reference
          "RFC 2710";
      }
      leaf report-v2-count {
        type yang:counter64;
        description
          "The number of Version 2 Multicast Listener Report.";
        reference
          "RFC 3810";
      }
      leaf done-count {
        type yang:counter64;
        description
          "The number of Version 1 Multicast Listener Done.";
        reference
          "RFC 2710";
      }
      leaf pim-hello-count {
        type yang:counter64;
        description
          "The number of PIM hello messages.";
        reference
          "RFC 7761";
      }
  } // mld-snooping-statistics

  augment "/rt:routing/rt:control-plane-protocols"+
        "/rt:control-plane-protocol" {
    when 'derived-from-or-self(rt:type, "ims:igmp-snooping")' {
      description
        "This container is only valid for IGMP snooping.";
    }
    description
      "IGMP snooping augmentation to control plane protocol
       configuration and state.";

    container igmp-snooping-instance {
      if-feature igmp-snooping;
      description
        "IGMP snooping instance to configure igmp-snooping.";

      leaf l2-service-type {
        type l2-service-type;
        default bridge;
```

```
        description
          "The l2-service-type indicates BRIDGE or L2VPN.";
      }

      uses instance-config-attributes-igmp-mld-snooping;

      leaf igmp-version {
        type uint8 {
          range "1..3";
        }
        default 2;
        description "IGMP version.";
      }

      leaf querier-source {
        type inet:ipv4-address;
        description
          "Use the IGMP snooping querier to support IGMP
           snooping in a VLAN where PIM and IGMP are not configured.
           The IPv4 address is used as source address in messages.";
      }

      list static-l2-multicast-group {
        if-feature static-l2-multicast-group;
        key "group source-addr";
        description
          "A static multicast route, (*,G) or (S,G).";

        leaf group {
          type rt-types:ipv4-multicast-group-address;
          description
            "Multicast group IPv4 address";
        }

        leaf source-addr {
          type rt-types:ipv4-multicast-source-address;
          description
            "Multicast source IPv4 address.";
        }

        leaf-list bridge-outgoing-interface {
          when 'derived-from-or-self(../../l2-service-
type,"ims:bridge")';
          type if:interface-ref;
          description "Outgoing interface in BRIDGE forwarding";
        }

        leaf-list l2vpn-outgoing-ac {
          when 'derived-from-or-self(../../l2-service-
type,"ims:l2vpn")';
          type if:interface-ref;
          description "Outgoing Attachment Circuit (AC) in L2VPN";
```

```
        }

        leaf-list l2vpn-outgoing-pw {
          when 'derived-from-or-self(../../l2-service-
type,"ims:l2vpn")';
          type pw:pseudowire-ref;
          description "Outgoing Pseudo Wire (PW) in L2VPN";
        }
      } // static-l2-multicast-group

      uses instance-state-attributes-igmp-mld-snooping;

      list group {

        key "address";

        config false;

        description "IGMP snooping information";

        leaf address {
          type rt-types:ipv4-multicast-group-address;
          description
            "Multicast group IPv4 address";
        }

        uses instance-state-group-attributes-igmp-mld-snooping;

        leaf last-reporter {
          type inet:ipv4-address;
          description
            "Address of the last host which has sent report to join
             the multicast group.";
        }

        list source {
          key "address";
          description "Source IPv4 address for multicast stream";

          leaf address {
            type rt-types:ipv4-multicast-source-address;
            description "Source IPv4 address for multicast stream";
          }

          uses instance-state-source-attributes-igmp-mld-snooping;

          leaf last-reporter {
            type inet:ipv4-address;
            description
              "Address of the last host which has sent report
               to join the multicast group.";
          }
```

```
          list host {
            if-feature explicit-tracking;
            key "host-address";
            description
              "List of multicast membership hosts
               of the specific multicast source-group.";

            leaf host-address {
              type inet:ipv4-address;
              description
                "Multicast membership host address.";
            }
            leaf host-filter-mode {
              type filter-mode-type;
              mandatory true;
              description
                "Filter mode for a multicast membership
                 host may be either include or exclude.";
            }
          }// list host

        } // list source
      } // list group

      container interfaces {
         config false;

         description
           "Contains the interfaces associated with the IGMP snooping
            instance";

         list interface {
           key "name";

           description
             "A list of interfaces  associated with the IGMP snooping
              instance";

           leaf name {
             type if:interface-ref;
             description
               "The name of interface";

           }

           container statistics {
             description
               "The interface statistics for IGMP snooping";

             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 received {
               description
                 "Number of received snooped IGMP packets";

               uses igmp-snooping-statistics;
             }
             container sent {
               description
                 "Number of sent snooped IGMP packets";

               uses igmp-snooping-statistics;
             }
           }
         }
       }

       action clear-igmp-snooping-groups {
         if-feature action-clear-groups;
         description
           "Clear IGMP snooping cache tables.";

         input {
           leaf group {
             type union {
               type enumeration {
                 enum 'all-groups' {
                   description
                     "All multicast group addresses.";
                 }
               }
               type rt-types:ipv4-multicast-group-address;
             }
             mandatory true;
             description
               "Multicast group IPv4 address. If value 'all-groups' is
                specified, all IGMP snooping group entries are cleared
                for specified source address.";
           }
           leaf source {
             type rt-types:ipv4-multicast-source-address;
             mandatory true;
             description
               "Multicast source IPv4 address. If value '*' is specified,
```

```
              all IGMP snooping source-group tables are cleared.";
          }
        }
      } // action clear-igmp-snooping-groups
    } // igmp-snooping-instance
  } // augment

  augment "/rt:routing/rt:control-plane-protocols"+
        "/rt:control-plane-protocol" {
    when 'derived-from-or-self(rt:type, "ims:mld-snooping")' {
        description
          "This container is only valid for MLD snooping.";
    }
    description
      "MLD snooping augmentation to control plane protocol
       configuration and state.";

    container mld-snooping-instance {
      if-feature mld-snooping;
      description
        "MLD snooping instance to configure mld-snooping.";

      leaf l2-service-type {
        type l2-service-type;
        default bridge;
        description
          "The l2-service-type indicates BRIDGE or L2VPN.";
      }

      uses instance-config-attributes-igmp-mld-snooping;

      leaf mld-version {
        type uint8 {
          range "1..2";
        }
        default 2;
        description "MLD version.";
      }

      leaf querier-source {
        type inet:ipv6-address;
        description
          "Use the MLD snooping querier to support MLD snooping where
           PIM and MLD are not configured. The IPv6 address is used as
           the source address in messages.";
      }

      list static-l2-multicast-group {
        if-feature static-l2-multicast-group;
        key "group source-addr";
        description
          "A static multicast route, (*,G) or (S,G).";
```

```
        leaf group {
          type rt-types:ipv6-multicast-group-address;
          description
            "Multicast group IPv6 address";
        }

        leaf source-addr {
          type rt-types:ipv6-multicast-source-address;
          description
            "Multicast source IPv6 address.";
        }

        leaf-list bridge-outgoing-interface {
          when 'derived-from-or-self(../../l2-service-
type,"ims:bridge")';
          type if:interface-ref;
          description "Outgoing interface in BRIDGE forwarding";
        }

        leaf-list l2vpn-outgoing-ac {
          when 'derived-from-or-self(../../l2-service-
type,"ims:l2vpn")';
          type if:interface-ref;
          description "Outgoing Attachment Circuit (AC) in L2VPN";
        }

        leaf-list l2vpn-outgoing-pw {
          when 'derived-from-or-self(../../l2-service-
type,"ims:l2vpn")';
          type pw:pseudowire-ref;
          description "Outgoing Pseudo Wire (PW) in L2VPN";
        }
      } // static-l2-multicast-group

      uses instance-state-attributes-igmp-mld-snooping;

      list group {
        key "address";
        config false;
        description "MLD snooping statistics information";

        leaf address {
          type rt-types:ipv6-multicast-group-address;
          description
            "Multicast group IPv6 address";
        }

        uses instance-state-group-attributes-igmp-mld-snooping;

        leaf last-reporter {
          type inet:ipv6-address;
```

```
          description
            "Address of the last host which has sent report
             to join the multicast group.";
        }

        list source {
          key "address";
          description "Source IPv6 address for multicast stream";

          leaf address {
            type rt-types:ipv6-multicast-source-address;
            description "Source IPv6 address for multicast stream";
          }

          uses instance-state-source-attributes-igmp-mld-snooping;

          leaf last-reporter {
           type inet:ipv6-address;
           description
            "Address of the last host which has sent report
             to join the multicast group.";
          }

          list host {
            if-feature explicit-tracking;
            key "host-address";
            description
              "List of multicast membership hosts
               of the specific multicast source-group.";

            leaf host-address {
              type inet:ipv6-address;
              description
                "Multicast membership host address.";
            }
            leaf host-filter-mode {
              type filter-mode-type;
              mandatory true;
              description
                "Filter mode for a multicast membership
                 host may be either include or exclude.";
            }
          }// list host
        } // list source
      } // list group

      container interfaces {
         config false;

         description
           "Contains the interfaces associated with the MLD snooping
            instance";
```

```
         list interface {
           key "name";

           description
             "A list of interfaces associated with the MLD snooping
              instance";

           leaf name {
             type if:interface-ref;
             description
               "The name of interface";

           }

           container statistics {
             description
               "The interface statistics for MLD snooping";

             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 received {
               description
                 "Number of received snooped MLD packets";

               uses mld-snooping-statistics;
             }
             container sent {
               description
                 "Number of sent snooped MLD packets";

               uses mld-snooping-statistics;
             }
           }
         }
      }

      action clear-mld-snooping-groups {
        if-feature action-clear-groups;
        description
          "Clear MLD snooping cache tables.";

        input {
```

```
          leaf group {
            type union {
              type enumeration {
                enum 'all-groups' {
                  description
                    "All multicast group addresses.";
                }
              }
              type rt-types:ipv6-multicast-group-address;
            }
            mandatory true;
            description
              "Multicast group IPv6 address. If value 'all-groups' is
               specified, all MLD snooping group entries are cleared
               for specified source address.";
          }
          leaf source {
            type rt-types:ipv6-multicast-source-address;
            mandatory true;
            description
              "Multicast source IPv6 address. If value '*' is specified,
               all MLD snooping source-group tables are cleared.";
          }
        }
      } // action clear-mld-snooping-groups
    }// mld-snooping-instance
  } // augment

  augment "/dot1q:bridges/dot1q:bridge" {
    description
      "Use IGMP & MLD snooping instance in BRIDGE.";

    leaf igmp-snooping-instance {
      type igmp-mld-snooping-instance-ref;
      description
        "Configure IGMP snooping instance under bridge view";
    }

    leaf mld-snooping-instance {
      type igmp-mld-snooping-instance-ref;
      description
        "Configure MLD snooping instance under bridge view";
    }
  }

  augment "/dot1q:bridges/dot1q:bridge"+
    "/dot1q:component/dot1q:bridge-vlan/dot1q:vlan" {
    description
      "Use IGMP & MLD snooping instance in certain VLAN of BRIDGE";

    leaf igmp-snooping-instance {
      type igmp-mld-snooping-instance-ref;
```

```
      description
        "Configure IGMP snooping instance under VLAN view";
    }

    leaf mld-snooping-instance {
      type igmp-mld-snooping-instance-ref;
      description
        "Configure MLD snooping instance under VLAN view";
    }
  }

  augment "/ni:network-instances/ni:network-instance"+
    "/ni:ni-type/l2vpn:l2vpn" {

    description
      "Use IGMP & MLD snooping instance in L2VPN.";

    leaf igmp-snooping-instance {
      type igmp-mld-snooping-instance-ref;
      description
        "Configure IGMP snooping instance in L2VPN.";
    }

    leaf mld-snooping-instance {
      type igmp-mld-snooping-instance-ref;
      description
        "Configure MLD snooping instance in L2VPN.";
    }
  }
}
<CODE ENDS>
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

Under /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:/

ims:igmp-snooping-instance

ims:mld-snooping-instance

The subtrees under /dot1q:bridges/dot1q:bridge

ims:igmp-snooping-instance

ims:mld-snooping-instance

The subtrees under /dot1q:bridges/dot1q:bridge/dot1q:component /dot1q:bridge-vlan/dot1q:vlan

ims:igmp-snooping-instance

ims:mld-snooping-instance

Unauthorized access to any data node of these subtrees can adversely affect the IGMP & MLD Snooping subsystem of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations, and other problems.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

Under /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:/

ims:igmp-snooping-instance

ims:mld-snooping-instance

Unauthorized access to any data node of these subtrees can disclose the operational state information of IGMP & MLD Snooping on this device. The group/source/host information may expose multicast group memberships, and transitively the associations between the user on the host and the contents from the source which could be privately sensitive. Some of the action operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

Under /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:/

ims:igmp-snooping-instance/ims:clear-igmp-snooping-groups

ims:mld-snooping-instance/ims:clear-mld-snooping-groups

Some of the actions in this YANG module may be considered sensitive or vulnerable in some network environments. The IGMP & MLD Snooping YANG module supports the "clear-igmp-snooping-groups" and "clear-mld-snooping-groups" actions. If unauthorized action is invoked, the IGMP and MLD Snooping group tables will be cleared unexpectedly. Especially when using wildcard, all the multicast traffic will be flooded in the broadcast domain. The devices that use this YANG module should heed the Security Considerations in [RFC4541].

6. IANA Considerations

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

6.1. XML Registry

This document registers the following namespace URIs in the IETF XML

registry [RFC3688]:

```
--------------------------------------------------------------------
URI: urn:ietf:params:xml:ns:yang:ietf-igmp-mld-snooping
Registrant Contact: The IETF.
XML: N/A, the requested URI is an XML namespace.
--------------------------------------------------------------------
```

6.2. YANG Module Names Registry

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

```
--------------------------------------------------------------------
name:         ietf-igmp-mld-snooping
namespace:    urn:ietf:params:xml:ns:yang:ietf-igmp-mld-snooping
prefix:       ims
reference:    RFC XXXX
--------------------------------------------------------------------
```

7. References

7.1. Normative References

[dot1Qcp] IEEE, "Standard for Local and metropolitan area networks--Bridges and Bridged Networks--Amendment 30: YANG Data Model", IEEE Std 802.1Qcp-2018 (Revision of IEEE Std 802.1Q-2014), September 2018, <https://ieeexplore.ieee.org/servlet/opac?punumber=8467505>

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

[RFC2236] W. Fenner, "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.

[RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.

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

[RFC4286] B. Haberman and J. Martin, "Multicast Router Discovery", RFC 4286, December 2005.

[RFC4541] M. Christensen, K. Kimball, F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, May 2006.

[RFC5790] H. Liu, W. Cao, H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, February 2010.

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

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

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

[RFC6636] H. Asaeda, H. Liu, Q. Wu, "Tuning the Behavior of the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) for Routers in Mobile and Wireless Networks", RFC 6636, May 2012.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, July 2013.

[RFC7761] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, R. Parekh, Z. Zhang, L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 7761, March 2016.

[RFC7950] M. Bjorklund, Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.

[RFC8040] A. Bierman, M. Bjorklund, K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

[RFC8294] X. Liu, Y. Qu, A. Lindem, C. Hopps, L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, December 2017.

[RFC8340] M. Bjorklund, and L. Berger, Ed., "YANG Tree Diagrams", RFC 8340, March 2018.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", RFC 8341, March 2018.

[RFC8342] M. Bjorklund and J. Schoenwaelder, "Network Management Datastore Architecture (NMDA)", RFC 8342, March 2018.

[RFC8343] M. Bjorklund, "A YANG Data Model for Interface Management", RFC 8343, March 2018.

[RFC8349] L. Lhotka, A. Lindem, Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, March 2018.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.

[RFC8529] L. Berger, C. Hopps, A. Lindem, D. Bogdanovic, X. Liu, "YANG Data Model for Network Instances", RFC 8529, March 2019.

[draft-ietf-bess-l2vpn-yang] Shah, H., Brissette, P., Chen, I., Hussain, I., Wen, B., and K. Tiruveedhula, "YANG Data Model for MPLS-basedL2VPN", draft-ietf-bess-l2vpn-yang-10 (work in progress), July 2019.

7.2. Informative References

[RFC3916] X. Xiao, Ed., D. McPherson, Ed., P. Pate, Ed., "Requirements for Pseudo-Wire Emulation Edge-to-Edge (PWE3)", RFC 3916, September 2004.

[RFC7951] L. Lhotka, "JSON Encoding of Data Modeled with YANG", RFC 7951, August 2016.

[RFC8407] A. Bierman, "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", RFC 8407, October 2018.

[RFC8652] X. Liu, F. Guo, M. Sivakumar, P. McAllister, A. Peter, "A YANG Data Model for the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)", RFC 8652, November 2019.

Appendix A. Data Tree Example

A.1 Bridge service

This section contains an example for bridge service in the JSON encoding [RFC7951], containing both configuration and state data.

```
              +-----------+
              +  Source   +
              +-----+-----+
                    |
   -----------------+----------------------------
                    |eth1/1
                +---+---+
                +  R1   +
                +-+---+-+
           eth1/2 |    \ eth1/3
                  |     \
                  |      \
                  |       \
                  |        \
           eth2/1 |         \ eth3/1
              +---+---+   +--+---+
              +  R2   +   +  R3  +
              +---+---+   +--+---+
           eth2/2 |          | eth3/2
                  |          |
   ---------------+----------+-------------------
                  |          |
                  |          |
         +--------+--+   +---+--------+
         + Receiver1 +   +  Receiver2 +
         +-----------+   +------------+
```

The configuration data for R1 in the above figure could be as follows:


```
{
"ietf-interfaces:interfaces":{
  "interface":[
   {
    "name":"eth1/1",
    "type":"iana-if-type:ethernetCsmacd"
   }
  ]
 },
 "ietf-routing:routing":{
  "control-plane-protocols":{
   "control-plane-protocol":[
    {
     "type":"ietf-igmp-mld-snooping:igmp-snooping",
     "name":"bis1",
     "ietf-igmp-mld-snooping:igmp-snooping-instance":{
```

```
       "l2-service-type":"ietf-igmp-mld-snooping:bridge",
       "enable":true
      }
     }
    ]
   }
  },
  "ieee802-dot1q-bridge:bridges":{
   "bridge":[
    {
     "name":"isp1",
     "address":"00-23-ef-a5-77-12",
     "bridge-type":"ieee802-dot1q-bridge:customer-vlan-bridge",
     "component":[
      {
       "name":"comp1",
       "type":"ieee802-dot1q-bridge:c-vlan-component",
       "bridge-vlan":{
        "vlan":[
         {
          "vid":101,
          "ietf-igmp-mld-snooping:igmp-snooping-instance":"bis1"
         }
        ]
       }
      }
     ]
    }
   ]
  }
}
```


The corresponding operational state data for R1 could be as follows:


```
{
 "ietf-interfaces:interfaces": {
  "interface": [
   {
    "name": "eth1/1",
    "type": "iana-if-type:ethernetCsmacd",
    "oper-status": "up",
    "statistics": {
     "discontinuity-time": "2018-05-23T12:34:56-05:00"
    }
   }
  ]
 },
 "ietf-routing:routing": {
  "control-plane-protocols": {
   "control-plane-protocol": [
    {
     "type": "ietf-igmp-mld-snooping:igmp-snooping",
```

```
     "name": "bis1",
     "ietf-igmp-mld-snooping:igmp-snooping-instance": {
      "l2-service-type": "ietf-igmp-mld-snooping:bridge",
      "enable": true
     }
    }
   ]
  }
 },
 "ieee802-dot1q-bridge:bridges": {
  "bridge": [
   {
    "name": "isp1",
    "address": "00-23-ef-a5-77-12",
    "bridge-type": "ieee802-dot1q-bridge:customer-vlan-bridge",
    "component": [
     {
      "name": "comp1",
      "type": "ieee802-dot1q-bridge:c-vlan-component",
      "bridge-vlan": {
       "vlan": [
        {
         "vid": 101,
         "ietf-igmp-mld-snooping:igmp-snooping-instance": "bis1"
        }
       ]
      }
     }
    ]
   }
  ]
 }
}
```


The following action is to clear all the entries whose group address is 225.1.1.1 for igmp-snooping-instance bis1.

```
POST /restconf/operations/ietf-routing:routing/control-plane-protocols/\
control-plane-protocol=ietf-igmp-mld-snooping:igmp-snooping,bis1/\
ietf-igmp-mld-snooping:igmp-snooping-instance/\
clear-igmp-snooping-groups HTTP/1.1
Host: example.com
Content-Type: application/yang-data+json

{
  "ietf-igmp-mld-snooping:input" : {
    "group": "225.1.1.1",
    "source": "*"
  }
}
```

A.2 L2VPN service

This section contains an example for L2VPN service in the JSON encoding [RFC7951], containing both configuration and state data.

```
              +-----------+
              +  Source   +
              +-----+-----+
                    |
   -----------------+----------------------------
                    |eth1/1
                +---+---+
                +  R1   +
                +-+---+-+
          eth1/2  |     \ eth1/3
                  |      \
                  |       \
                  |        \
                  |         \
          eth2/1  |          \ eth3/1
              +---+---+    +-+---+
              +  R2   +----+ R3  +
              +---+---+    +-+---+
          eth2/2  |          | eth3/2
                  |          |
   ---------------+----------+-------------------
                  |          |
                  |          |
         +--------+--+   +---+--------+
         + Receiver1 +   +  Receiver2 +
         +-----------+   +------------+
```

The configuration data for R1 in the above figure could be as follows:


```
{
"ietf-interfaces:interfaces":{
  "interface":[
   {
    "name":"eth1/1",
    "type":"iana-if-type:ethernetCsmacd"
   }
  ]
 },
 "ietf-pseudowires:pseudowires": {
  "pseudowire": [
   {
    "name": "pw2"
   },
   {
    "name": "pw3"
```

```
    }
   ]
  },
  "ietf-network-instance:network-instances": {
   "network-instance": [
    {
     "name": "vpls1",
     "ietf-igmp-mld-snooping:igmp-snooping-instance": "vis1",
     "ietf-l2vpn:type": "ietf-l2vpn:vpls-instance-type",
     "ietf-l2vpn:signaling-type": "ietf-l2vpn:ldp-signaling",
     "ietf-l2vpn:endpoint": [
      {
       "name": "acs",
       "ac": [
         {
          "name": "eth1/1"
         }
       ]
      },
      {
       "name": "pws",
       "pw": [
         {
          "name": "pw2"
         },
         {
          "name": "pw3"
         }
       ]
      }
     ]
    }
   ]
  },
  "ietf-routing:routing": {
   "control-plane-protocols": {
    "control-plane-protocol": [
     {
      "type": "ietf-igmp-mld-snooping:igmp-snooping",
      "name": "vis1",
      "ietf-igmp-mld-snooping:igmp-snooping-instance": {
       "l2-service-type": "ietf-igmp-mld-snooping:l2vpn",
       "enable": true
      }
     }
    ]
   }
  }
}
```

The corresponding operational state data for R1 could be as follows:


```
{
"ietf-interfaces:interfaces":{
  "interface":[
   {
    "name":"eth1/1",
    "type":"iana-if-type:ethernetCsmacd",
    "oper-status": "up",
    "statistics": {
     "discontinuity-time": "2018-05-23T12:34:56-05:00"
    }
   }
  ]
 },
 "ietf-pseudowires:pseudowires": {
  "pseudowire": [
   {
    "name": "pw2"
   },
   {
    "name": "pw3"
   }
  ]
 },
 "ietf-network-instance:network-instances": {
  "network-instance": [
   {
    "name": "vpls1",
    "ietf-igmp-mld-snooping:igmp-snooping-instance": "vis1",
    "ietf-l2vpn:type": "ietf-l2vpn:vpls-instance-type",
    "ietf-l2vpn:signaling-type": "ietf-l2vpn:ldp-signaling",
    "ietf-l2vpn:endpoint": [
     {
      "name": "acs",
      "ac": [
        {
         "name": "eth1/1"
        }
      ]
     },
     {
      "name": "pws",
      "pw": [
        {
         "name": "pw2"
        },
        {
         "name": "pw3"
        }
      ]
     }
```

```
    ]
   }
  ]
 },
 "ietf-routing:routing": {
  "control-plane-protocols": {
   "control-plane-protocol": [
    {
     "type": "ietf-igmp-mld-snooping:igmp-snooping",
     "name": "vis1",
     "ietf-igmp-mld-snooping:igmp-snooping-instance": {
      "l2-service-type": "ietf-igmp-mld-snooping:l2vpn",
      "enable": true
     }
    }
   ]
  }
 }
}
```

Authors' Addresses

Hongji Zhao
Ericsson (China) Communications Company Ltd.
Ericsson Tower, No. 5 Lize East Street,
Chaoyang District Beijing 100102, P.R. China

Email: hongji.zhao@ericsson.com

Xufeng Liu
Volta Networks
USA

EMail: xufeng.liu.ietf@gmail.com

Yisong Liu
China Mobile
China

Email: liuyisong@chinamobile.com

Anish Peter
Individual

EMail: anish.ietf@gmail.com

Mahesh Sivakumar
Juniper Networks
1133 Innovation Way
Sunnyvale, California
USA

EMail: sivakumar.mahesh@gmail.com

PIM Working Group X. Liu
Internet-Draft Volta Networks
Intended Status: Standard Track F. Guo
Expires: December 14, 2019 Huawei
M. Sivakumar
Juniper
P. McAllister
Metaswitch Networks
A. Peter
Individual
June 14, 2019

A YANG Data Model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)
draft-ietf-pim-igmp-mld-yang-15

Status of this Memo

Copyright Notice

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.

Conventions used in this document

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

Table of Contents

1. Introduction

YANG [RFC6020] [RFC7950] is a data definition language that was introduced to model the configuration and running state of a device managed using network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. YANG is now also being used as a component of wider management interfaces, such as command line interfaces (CLIs).

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. The protocol versions include IGMPv1 [RFC1112], IGMPv2 [RFC2236], IGMPv3 [RFC3376], MLDv1 [RFC2710], and MLDv2 [RFC3810]. The core features of the IGMP and MLD protocols are defined as required. Non-core features are defined as optional in the provided data model.

The YANG model in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342].

1.1. Terminology

The terminology for describing YANG data models is found in [RFC6020] and [RFC7950], including:

- augment
- data model
- data node
- identity
- module

The following abbreviations are used in this document and the defined model:

IGMP:

Internet Group Management Protocol [RFC3376].

MLD:

Multicast Listener Discovery [RFC3810].

SSM:

Source-Specific Multicast service model [RFC3569] [RFC4607].

1.2. Tree Diagrams

Tree diagrams used in this document follow the notation defined in [RFC8340].

1.3. Prefixes in Data Node Names

In this document, names of data nodes, actions, and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
if	ietf-interfaces	[RFC8343]
ip	ietf-ip	[RFC8344]
rt	ietf-routing	[RFC8349]
rt-types	ietf-routing-types	[RFC8294]
acl	ietf-access-control-list	[RFC8519]

Table 1: Prefixes and Corresponding YANG Modules

2. Design of Data model

2.1. Scope of Model

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

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

This model can be used to configure and manage various versions of IGMP and MLD protocols. The operational state data and statistics can be retrieved by this model. Even though no protocol specific notifications are defined in this model, the subscription and push mechanism defined in [I-D.ietf-netconf-subscribed-notifications] and [I-D.ietf-netconf-yang-push] can be used by the user to subscribe to notifications on the data nodes in this model.

The model contains all the basic configuration parameters to operate the protocols listed above. Depending on the implementation choices, some systems may not allow some of the advanced parameters to be configurable. The occasionally implemented parameters are modeled as optional features in this model, while the rarely implemented parameters are not included this model and left for augmentation. This model can be extended, and has been structured in a way that such extensions can be conveniently made.

The protocol parameters covered in this model can been seen from the model structure described in Section 3.

The protocol parameters that were considered but are not covered in this model are described in the following sections.

2.1.1. Parameters Not Covered at Global Level

The configuration parameters and operational states not covered on an IGMP instance or an MLD instance are:

- o Explicit tracking
- o Maximum transmit rate
- o Last member query count
- o Other querier present time
- o Send router alert
- o Startup query interval
- o Startup query count

2.1.2. Parameters Not Covered at Interface Level

The configuration parameters and operational states not covered on an IGMP interface or an MLD interface are:

- o Disable router alert check
- o Drop IGMP version 1, IGMP version 2, or MLD version 1
- o Last member query count
- o Maximum number of sources
- o Other querier present time
- o Passive mode
- o Promiscuous mode

o Query before immediate leave

o Send router alert

2.2. Optional Capabilities

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

There is also value in widely-supported features being standardized, to provide a standardized way to access these features, to save work for individual vendors, and so that mapping between different vendors' configuration is not needlessly complicated. Therefore this model declares 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.

2.3. Position of Address Family in Hierarchy

The protocol IGMP only supports IPv4, while the protocol MLD only supports IPv6. The data model defined in this document can be used for both IPv4 and IPv6 address families.

This document defines IGMP and MLD as separate schema branches in the structure. The benefits are:

o The model can support IGMP (IPv4), MLD (IPv6), or both optionally and independently. Such flexibility cannot be achieved cleanly with a combined branch.

o The structure is consistent with other YANG models such as RFC 8344, which uses separate branches for IPv4 and IPv6.

o The separate branches for IGMP and MLD can accommodate their differences better and cleaner. The two branches can better support different features and node types.

3. Module Structure

This model augments the core routing data model specified in [RFC8349].

```
      +--rw routing
         +--rw router-id?
         +--rw control-plane-protocols
         |  +--rw control-plane-protocol* [type name]
         |     +--rw type
         |     +--rw name
         |     +--rw igmp     <= Augmented by this Model
                     ...
         |     +--rw mld      <= Augmented by this Model
                     ...
```

The "igmp" container instantiates an IGMP protocol of version IGMPv1, IGMPv2, or IGMPv3. The "mld" container instantiates an MLD protocol of version MLDv1 or MLDv2.

The YANG data model defined in this document conforms to the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [RFC8407].

A configuration data node is marked as mandatory only when its value must be provided by the user. Where nodes are not essential to protocol operation, they are marked as optional. Some other nodes are essential but have a default specified, so that they are also optional and need not be configured explicitly.

3.1. IGMP Configuration and Operational State

The IGMP data is modeled as a schema subtree augmenting the "control-plane-protocol" data node under "/rt:routing/rt:control-plane-protocols" in the module ietf-routing, following the convention described in [RFC8349]. The augmentation to the module ietf-routing allows this model to support multiple instances of IGMP, but a restriction MAY be added depending on the implementation and the device. The identity "igmp" is derived from the "rt:control-plane-protocol" base identity and indicates that a control-plane-protocol instance is IGMP.

The IGMP subtree is a three-level hierarchy structure as listed below:

Global level: Including IGMP configuration and operational state attributes for the entire IGMP protocol instance in this router.

Interface-global level: Including configuration data nodes that are applicable to all the interfaces whose corresponding nodes are not defined or not configured at the interface level. For such a node at the interface level, the system uses the same value of the corresponding node at the interface-global level.

Interface level: Including IGMP configuration and operational state attributes specific to the given interface. For a configuration node at the interface level, there may exist a corresponding configuration node with the same name at the interface-global level. The value configured on a node at the interface level overrides the value configured on the corresponding node at the interface-global level.

```
augment /rt:routing/rt:control-plane-protocols
          /rt:control-plane-protocol:
  +--rw igmp {feature-igmp}?
     +--rw global
     |  +--rw enable?          boolean {global-admin-enable}?
     |  +--rw max-entries?     uint32 {global-max-entries}?
     |  +--rw 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
     +--rw interfaces
        +--rw last-member-query-interval?   uint16
        +--rw query-interval?               uint16
        +--rw query-max-response-time?      uint16
```

```
             +--rw require-router-alert?         boolean
             |       {intf-require-router-alert}?
             +--rw robustness-variable?          uint8
             +--rw version?                      uint8
             +--rw max-groups-per-interface?     uint32
             |       {intf-max-groups}?
             +--rw interface* [interface-name]
                +--rw interface-name                 if:interface-ref
                +--rw last-member-query-interval?   uint16
                +--rw query-interval?                uint16
                +--rw query-max-response-time?      uint16
                +--rw require-router-alert?         boolean
                |       {intf-require-router-alert}?
                +--rw robustness-variable?          uint8
                +--rw version?                      uint8
                +--rw enable?                       boolean
                |       {intf-admin-enable}?
                +--rw group-policy?
                |       -> /acl:acls/acl/name
                +--rw immediate-leave?              empty
                |       {intf-immediate-leave}?
                +--rw max-groups?                   uint32
                |       {intf-max-groups}?
                +--rw max-group-sources?            uint32
                |       {intf-max-group-sources}?
                +--rw source-policy?
                |       -> /acl:acls/acl/name {intf-source-policy}?
                +--rw verify-source-subnet?         empty
                |       {intf-verify-source-subnet}?
                +--rw explicit-tracking?            empty
                |       {intf-explicit-tracking}?
                +--rw exclude-lite?                 empty
                |       {intf-exclude-lite}?
                +--rw join-group*
                |       rt-types:ipv4-multicast-group-address
                |       {intf-join-group}?
                +--rw ssm-map*
                |  |    [ssm-map-source-addr ssm-map-group-policy]
                |  |    {intf-ssm-map}?
                |  +--rw ssm-map-source-addr     ssm-map-ipv4-addr-type
                |  +--rw ssm-map-group-policy    string
                +--rw static-group* [group-addr source-addr]
                |  |    {intf-static-group}?
                |  +--rw group-addr
                |  |       rt-types:ipv4-multicast-group-address
                |  +--rw source-addr
                |          rt-types:ipv4-multicast-source-address
                +--ro oper-status                   enumeration
                +--ro querier                       inet:ipv4-address
```

```
                +--ro joined-group*
                |       rt-types:ipv4-multicast-group-address
                |       {intf-join-group}?
                +--ro group* [group-address]
                   +--ro group-address
                   |       rt-types:ipv4-multicast-group-address
                   +--ro expire           uint32
                   +--ro filter-mode      enumeration
                   +--ro up-time          uint32
                   +--ro last-reporter?   inet:ipv4-address
                   +--ro source* [source-address]
                      +--ro source-address    inet:ipv4-address
                      +--ro expire            uint32
                      +--ro up-time           uint32
                      +--ro host-count?       uint32
                      |       {intf-explicit-tracking}?
                      +--ro last-reporter?    inet:ipv4-address
                      +--ro host* [host-address]
                         |    {intf-explicit-tracking}?
                         +--ro host-address        inet:ipv4-address
                         +--ro host-filter-mode    enumeration
```

3.2. MLD Configuration and Operational State

The MLD data is modeled as a schema subtree augmenting the "control-plane-protocol" data node under "/rt:routing/rt:control-plane-protocols" in the module ietf-routing, following the convention described in [RFC8349]. The augmentation to the module ietf-routing allows this model to support multiple instances of MLD, but a restriction MAY be added depending on the implementation and the device. The identity "mld" is derived from the "rt:control-plane-protocol" base identity and indicates that a control-plane-protocol instance is MLD.

The MLD subtree is a three-level hierarchy structure as listed below:

Global level: Including MLD configuration and operational state attributes for the entire MLD protocol instance in this router.

Interface-global level: Including configuration data nodes that are applicable to all the interfaces whose corresponding nodes are not defined or not configured at the interface level. For such a node at the interface level, the system uses the same value of the corresponding node at the interface-global level.

Interface level: Including MLD configuration and operational state attributes specific to the given interface. For a configuration node at the interface level, there may exist a corresponding configuration node with the same name at the interface-global level. The value configured on a node at the interface level overrides the value configured on the corresponding node at the interface-global level.

```
  augment /rt:routing/rt:control-plane-protocols
            /rt:control-plane-protocol:
    +--rw mld {feature-mld}?
       +--rw global
       |  +--rw enable?          boolean {global-admin-enable}?
       |  +--rw max-entries?     uint32 {global-max-entries}?
       |  +--rw 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
       +--rw interfaces
          +--rw last-member-query-interval?   uint16
          +--rw query-interval?               uint16
          +--rw query-max-response-time?      uint16
          +--rw require-router-alert?         boolean
          |       {intf-require-router-alert}?
          +--rw robustness-variable?          uint8
          +--rw version?                      uint8
          +--rw max-groups-per-interface?     uint32
          |       {intf-max-groups}?
          +--rw interface* [interface-name]
             +--rw interface-name                if:interface-ref
             +--rw last-member-query-interval?   uint16
             +--rw query-interval?               uint16
             +--rw query-max-response-time?      uint16
             +--rw require-router-alert?         boolean
```

```
                 |       {intf-require-router-alert}?
                 +--rw robustness-variable?          uint8
                 +--rw version?                      uint8
                 +--rw enable?                       boolean
                 |       {intf-admin-enable}?
                 +--rw group-policy?
                 |       -> /acl:acls/acl/name
                 +--rw immediate-leave?              empty
                 |       {intf-immediate-leave}?
                 +--rw max-groups?                   uint32
                 |       {intf-max-groups}?
                 +--rw max-group-sources?            uint32
                 |       {intf-max-group-sources}?
                 +--rw source-policy?
                 |       -> /acl:acls/acl/name {intf-source-policy}?
                 +--rw verify-source-subnet?         empty
                 |       {intf-verify-source-subnet}?
                 +--rw explicit-tracking?            empty
                 |       {intf-explicit-tracking}?
                 +--rw exclude-lite?                 empty
                 |       {intf-exclude-lite}?
                 +--rw join-group*
                 |       rt-types:ipv6-multicast-group-address
                 |       {intf-join-group}?
                 +--rw ssm-map*
                 |  |    [ssm-map-source-addr ssm-map-group-policy]
                 |  |    {intf-ssm-map}?
                 |  +--rw ssm-map-source-addr     ssm-map-ipv6-addr-type
                 |  +--rw ssm-map-group-policy    string
                 +--rw static-group* [group-addr source-addr]
                 |  |    {intf-static-group}?
                 |  +--rw group-addr
                 |  |       rt-types:ipv6-multicast-group-address
                 |  +--rw source-addr
                 |          rt-types:ipv6-multicast-source-address
                 +--ro oper-status                   enumeration
                 +--ro querier                       inet:ipv6-address
                 +--ro joined-group*
                 |       rt-types:ipv6-multicast-group-address
                 |       {intf-join-group}?
                 +--ro group* [group-address]
                    +--ro group-address
                    |       rt-types:ipv6-multicast-group-address
                    +--ro expire           uint32
                    +--ro filter-mode      enumeration
                    +--ro up-time          uint32
                    +--ro last-reporter?   inet:ipv6-address
                    +--ro source* [source-address]
                       +--ro source-address    inet:ipv6-address
```

```
                      +--ro expire            uint32
                      +--ro up-time           uint32
                      +--ro host-count?       uint32
                      |       {intf-explicit-tracking}?
                      +--ro last-reporter?    inet:ipv6-address
                      +--ro host* [host-address]
                         |    {intf-explicit-tracking}?
                         +--ro host-address        inet:ipv6-address
                         +--ro host-filter-mode    enumeration
```

3.3. IGMP and MLD Actions

IGMP and MLD each have one action which clears the group membership cache entries for that protocol.

```
     augment /rt:routing/rt:control-plane-protocols
               /rt:control-plane-protocol:
       +--rw igmp {feature-igmp}?
          +---x clear-groups {action-clear-groups}?
             +---w input
                +---w (interface)
                |  +--:(name)
                |  |  +---w interface-name?   leafref
                |  +--:(all)
                |     +---w all-interfaces?   empty
                +---w group-address           union
                +---w source-address
                        rt-types:ipv4-multicast-source-address

     augment /rt:routing/rt:control-plane-protocols
               /rt:control-plane-protocol:
       +--rw mld {feature-mld}?
          +---x clear-groups {action-clear-groups}?
             +---w input
                +---w (interface)
                |  +--:(name)
                |  |  +---w interface-name?   leafref
                |  +--:(all)
                |     +---w all-interfaces?   empty
                +---w group-address?          union
                +---w source-address?
                        rt-types:ipv6-multicast-source-address
```

4. IGMP and MLD YANG Module

This module references [RFC1112], [RFC2236], [RFC2710], [RFC3376], [RFC3810], [RFC5790], [RFC6636], [RFC6991], [RFC8294], [RFC8343], [RFC8344], [RFC8349], and [RFC8519].

```
<CODE BEGINS> file "ietf-igmp-mld@2019-06-12.yang"
module ietf-igmp-mld {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld";
  prefix igmp-mld;

  import ietf-inet-types {
    prefix "inet";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-yang-types {
    prefix "yang";
    reference "RFC 6991: Common YANG Data Types";
  }

  import ietf-routing-types {
    prefix "rt-types";
    reference
      "RFC 8294: Common YANG Data Types for the Routing Area";
  }

  import ietf-access-control-list {
    prefix "acl";
    reference
      "RFC 8519: YANG Data Model for Network Access Control Lists
       (ACLs)";
  }

  import ietf-routing {
    prefix "rt";
    reference
      "RFC 8349: A YANG Data Model for Routing Management (NMDA
       Version)";
  }

  import ietf-interfaces {
    prefix "if";
    reference "RFC 8343: A YANG Data Model for Interface Management";
  }

  import ietf-ip {
    prefix ip;
    reference "RFC 8344: A YANG Data Model for IP Management";
  }

  organization
    "IETF PIM Working Group";
```

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

        Editor:   Xufeng Liu
                  <mailto:xufeng.liu.ietf@gmail.com>

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

        Editor:   Mahesh Sivakumar
                  <mailto:sivakumar.mahesh@gmail.com>

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

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

     description
       "The module defines the configuration and operational state for
        the Internet Group Management Protocol (IGMP) and Multicast
        Listener Discovery (MLD) protocols.

        Copyright (c) 2019 IETF Trust and the persons identified as
        authors of the code.  All rights reserved.

        Redistribution and use in source and binary forms, with or
        without modification, is permitted pursuant to, and subject to
        the license terms contained in, the Simplified BSD License set
        forth in Section 4.c of the IETF Trust's Legal Provisions
        Relating to IETF Documents
        (http://trustee.ietf.org/license-info).

        This version of this YANG module is part of RFC XXXX; see the
        RFC itself for full legal notices.";

     // RFC Ed.: replace XXXX with actual RFC number and remove
     // this note
     revision 2019-06-12 {
       description
         "Initial revision.";
       reference
         "RFC XXXX: A YANG Data Model for IGMP and MLD";
     }

     /*
      * Features
      */
```

```
     feature feature-igmp {
       description
         "Support IGMP protocol for IPv4 group membership record.";
     }

     feature feature-mld {
       description
         "Support MLD protocol for IPv6 group membership record.";
     }

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

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

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

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

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

     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 intf-explicit-tracking {
       description
         "Support configuration of interface explicit-tracking hosts.";
     }

     feature intf-lite-exclude-filter {
       description
         "Support configuration of interface lite-exclude-filter.";
     }

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

     feature action-clear-groups {
       description
         "Support actions 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

     /*
      * Identities
      */
     identity igmp {
       base "rt:control-plane-protocol";
       description "IGMP protocol.";
       reference
         "RFC 3376: Internet Group Management Protocol, Version 3.";
     }

     identity mld {
       base "rt:control-plane-protocol";
       description "MLD protocol.";
       reference
         "RFC 3810: Multicast Listener Discovery Version 2 (MLDv2) for
          IPv6.";
```

```
     }

     /*
      * Groupings
      */
     grouping global-config-attributes {
       description
         "This grouping is used in either IGMP schema or MLD schema.
          When used in IGMP schema, this grouping contains the global
          configuration for IGMP;
          when used in MLD schema, this grouping contains the global
          configuration for MLD.";

       leaf enable {
         if-feature global-admin-enable;
         type boolean;
         default true;
         description
           "When this grouping is used for IGMP, this leaf indicates
            whether IGMP is enabled ('true') or disabled ('false')
            in the routing instance.
            When this grouping is used for MLD, this leaf indicates
            whether MLD is enabled ('true') or disabled ('false')
            in the routing instance.";
       }
       leaf max-entries {
         if-feature global-max-entries;
         type uint32;
         description
           "When this grouping is used for IGMP, this leaf indicates
            the maximum number of entries in the IGMP instance.
            When this grouping is used for MLD, this leaf indicates
            the maximum number of entries in the MLD instance.
            If this leaf is not specified, the number of entries is not
            limited.";
       }
       leaf max-groups {
         if-feature global-max-groups;
         type uint32;
         description
           "When this grouping is used for IGMP, this leaf indicates
            the maximum number of groups in the IGMP instance.
            When this grouping is used for MLD, this leaf indicates
            the maximum number of groups in the MLD instance.
            If this leaf is not specified, the number of groups is not
            limited.";
       }
     } // global-config-attributes
```

```
     grouping global-state-attributes {
       description
         "This grouping is used in either IGMP schema or MLD schema.
          When used in IGMP schema, this grouping contains the global
          IGMP state attributes;
          when used in MLD schema, this grouping contains the global
          MLD state attributes;";

       leaf entries-count {
         type uint32;
         config false;
         description
           "When this grouping is used for IGMP, this leaf indicates
            the number of entries in the IGMP instance.
            When this grouping is used for MLD, this leaf indicates
            the number of entries in the MLD instance.";
       }
       leaf groups-count {
         type uint32;
         config false;
         description
           "When this grouping is used for IGMP, this leaf indicates
            the number of existing groups in the IGMP instance.
            When this grouping is used for MLD, this leaf indicates
            the number of existing groups in the MLD instance.";
       }

       container statistics {
         config false;
         description
           "When this grouping is used for IGMP, this container contains
            the statistics for the IGMP instance.
            When this grouping is used for MLD, this leaf indicates
            the statistics for the MLD instance.";

         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 interface-global-config-attributes {
       description
         "Configuration attributes applied to the interface-global level
          whose per interface attributes are not configured.";

       leaf max-groups-per-interface {
         if-feature intf-max-groups;
         type uint32;
         description
           "The maximum number of groups associated with each interface.
            If this leaf is not specified, the number of groups is not
            limited.";
       }
     } //interface-global-config-attributes

     grouping interface-common-config-attributes {
       description
         "Configuration attributes applied to both the interface-global
          level and interface level.";

       leaf last-member-query-interval {
         type uint16 {
           range "1..1023";
         }
         units seconds;
         description
           "When used in IGMP schema, this leaf indicates the Last
            Member Query Interval, which may be tuned to modify the
            leave latency of the network;
            when used in MLD schema, this leaf indicates the Last
            Listener Query Interval, which may be tuned to modify the
            leave latency of the network.
            This leaf is not applicable for version 1 of the IGMP. For
            version 2 and version 3 of the IGMP, and for all versions of
            the MLD, the default value of this leaf is 1.
            This leaf may be configured at the interface level or the
            interface-global level, with precedence given to the value
            at the interface level. If the leaf is not configured at
            either level, the default value is used.";
         reference
           "RFC 2236. Sec. 8.8. RFC 3376. Sec. 8.8.
            RFC 2710. Sec. 7.8. RFC 3810. Sec. 9.8.";
       }
       leaf query-interval {
```

```
         type uint16 {
           range "1..31744";
         }
         units seconds;
         description
           "The Query Interval is the interval between General Queries
            sent by the Querier. In RFC 3376, the Querier's Query
            Interval(QQI) is represented from the Querier's Query
            Interval Code in query message as follows:
            If QQIC < 128, QQI = QQIC.
            If QQIC >= 128, QQIC represents a floating-point value as
            follows:
             0 1 2 3 4 5 6 7
            +-+-+-+-+-+-+-+-+
            |1| exp | mant  |
            +-+-+-+-+-+-+-+-+
            QQI = (mant | 0x10) << (exp + 3).
            The maximum value of QQI is 31744.
            The default value is 125.
            This leaf may be configured at the interface level or the
            interface-global level, with precedence given to the value
            at the interface level. If the leaf is not configured at
            either level, the default value is used.";
         reference "RFC 3376. Sec. 4.1.7, 8.2, 8.14.2.";
       }
       leaf query-max-response-time {
         type uint16 {
           range "1..1023";
         }
         units seconds;
         description
           "Query maximum response time specifies the maximum time
            allowed before sending a responding report.
            The default value is 10.
            This leaf may be configured at the interface level or the
            interface-global level, with precedence given to the value
            at the interface level. If the leaf is not configured at
            either level, the default value is used.";
         reference "RFC 3376. 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.
            When this leaf is not configured, the server uses the
            following rules to determine the operational value of this
            leaf:
            if this grouping is used in IGMP schema and the value of the
```

```
            leaf 'version' is 1, the value 'false' is operationally used
            by the server;
            if this grouping is used in IGMP schema and the value of the
            leaf 'version' is 2 or 3, the value 'true' is operationally
            used by the server;
            if this grouping is used in MLD schema, the value 'true' is
            operationally used by the server.
            This leaf may be configured at the interface level or the
            interface-global level, with precedence given to the value
            at the interface level. If the leaf is not configured at
            either level, the default value is used.";
       }
       leaf robustness-variable {
         type uint8 {
           range "1..7";
         }
         description
           "Querier's Robustness Variable allows tuning for the
            expected packet loss on a network.
            The default value is 2.
            This leaf may be configured at the interface level or the
            interface-global level, with precedence given to the value
            at the interface level. If the leaf is not configured at
            either level, the default value is used.";
         reference "RFC 3376. Sec. 4.1.6, 8.1, 8.14.1.";
       }
     } // interface-common-config-attributes

     grouping interface-common-config-attributes-igmp {
       description
         "Configuration attributes applied to both the interface-global
          level and interface level for IGMP.";

       uses interface-common-config-attributes;
       leaf version {
         type uint8 {
           range "1..3";
         }
         description
           "IGMP version.
            The default value is 2.
            This leaf may be configured at the interface level or the
            interface-global level, with precedence given to the value
            at the interface level. If the leaf is not configured at
            either level, the default value is used.";
         reference "RFC 1112, RFC 2236, RFC 3376.";
       }
     }
```

```
     grouping interface-common-config-attributes-mld {
       description
         "Configuration attributes applied to both the interface-global
          level and interface level for MLD.";

       uses interface-common-config-attributes;
       leaf version {
         type uint8 {
           range "1..2";
         }
         description
           "MLD version.
            The default value is 2.
            This leaf may be configured at the interface level or the
            interface-global level, with precedence given to the value
            at the interface level. If the leaf is not configured at
            either level, the default value is used.";
         reference "RFC 2710, RFC 3810.";
       }
     }

     grouping interfaces-config-attributes-igmp {
       description
         "Configuration attributes applied to the interface-global
          level for IGMP.";

       uses interface-common-config-attributes-igmp;
       uses interface-global-config-attributes;
     }

     grouping interfaces-config-attributes-mld {
       description
         "Configuration attributes applied to the interface-global
          level for MLD.";

       uses interface-common-config-attributes-mld;
       uses interface-global-config-attributes;
     }

     grouping interface-level-config-attributes {
       description
         "This grouping is used in either IGMP schema or MLD schema.
          When used in IGMP schema, this grouping contains the IGMP
          configuration attributes that are defined at the interface
          level but are not defined at the interface-global level;
          when used in MLD schema, this grouping contains the MLD
          configuration attributes that are defined at the interface
          level but are not defined at the interface-global level.";
```

```
       leaf enable {
         if-feature intf-admin-enable;
         type boolean;
         default true;
         description
           "When this grouping is used for IGMP, this leaf indicates
            whether IGMP is enabled ('true') or disabled ('false')
            on the interface.
            When this grouping is used for MLD, this leaf indicates
            whether MLD is enabled ('true') or disabled ('false')
            on the interface.";
       }
       leaf group-policy {
         type leafref {
           path "/acl:acls/acl:acl/acl:name";
         }
         description
           "When this grouping is used for IGMP, this leaf specifies
            the name of the access policy used to filter the
            IGMP membership.
            When this grouping is used for MLD, this leaf specifies
            the name of the access policy used to filter the
            MLD membership.
            The value space of this leaf is restricted to the existing
            policy instances defined by the referenced schema RFC 8519.
            As specified by RFC 8519, the length of the name is between
            1 and 64; a device MAY further restrict the length of this
            name; space and special characters are not allowed.
            If this leaf is not specified, no policy is applied, and
            all packets received from this interface are accepted.";
         reference
           "RFC 8519: YANG Data Model for Network Access Control Lists
            (ACLs)";
       }
       leaf immediate-leave {
         if-feature intf-immediate-leave;
         type empty;
         description
           "When this grouping is used for IGMP, the presence of this
            leaf requests IGMP to perform an immediate leave upon
            receiving an 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
            the group.
            When this grouping is used for MLD, the presence of this
            leaf requests MLD to perform an immediate leave upon
            receiving an MLDv1 leave message.
            If the router is MLD-enabled, it sends an MLD last member
```

```
            query with a last member query response time. However, the
            router does not wait for the response time before it prunes
            the group.";
       }
       leaf max-groups {
         if-feature intf-max-groups;
         type uint32;
         description
           "When this grouping is used for IGMP, this leaf indicates
            the maximum number of groups associated with the IGMP
            interface.
            When this grouping is used for MLD, this leaf indicates
            the maximum number of groups associated with the MLD
            interface.
            If this leaf is not specified, the number of groups is not
            limited.";
       }
       leaf max-group-sources {
         if-feature intf-max-group-sources;
         type uint32;
         description
           "The maximum number of group sources.
            If this leaf is not specified, the number of group sources
            is not limited.";
       }
       leaf source-policy {
         if-feature intf-source-policy;
         type leafref {
           path "/acl:acls/acl:acl/acl:name";
         }
         description
           "Name of the access policy used to filter sources.
            The value space of this leaf is restricted to the existing
            policy instances defined by the referenced schema RFC 8519.
            As specified by RFC 8519, the length of the name is between
            1 and 64; a device MAY further restrict the length of this
            name; space and special characters are not allowed.
            If this leaf is not specified, no policy is applied, and
            all packets received from this interface are accepted.";
       }
       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 explicit-tracking {
         if-feature intf-explicit-tracking;
```

```
         type empty;
         description
           "When this grouping is used for IGMP, the presence of this
            leaf enables IGMP-based explicit membership tracking
            function for multicast routers and IGMP proxy devices
            supporting IGMPv3.
            When this grouping is used for MLD, the presence of this
            leaf enables MLD-based explicit membership tracking
            function for multicast routers and MLD proxy devices
            supporting MLDv2.
            The explicit membership tracking function contributes to
            saving network resources and shortening leave latency.";
         reference
           "RFC 6636. Sec 3.";
       }
       leaf lite-exclude-filter {
         if-feature intf-lite-exclude-filter;
         type empty;
         description
           "When this grouping is used for IGMP, the presence of this
            leaf enables the support of the simplified EXCLUDE filter
            in the Lightweight IGMPv3 protocol, which simplifies the
            standard versions of IGMPv3.
            When this grouping is used for MLD, the presence of this
            leaf enables the support of the simplified EXCLUDE filter
            in the Lightweight MLDv2 protocol, which simplifies the
            standard versions of MLDv2.";
         reference "RFC 5790";
       }
     } // interface-level-config-attributes

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

       uses interface-common-config-attributes-igmp;
       uses interface-level-config-attributes;
       leaf-list join-group {
         if-feature intf-join-group;
         type rt-types:ipv4-multicast-group-address;
         description
           "The router joins this multicast group on the interface.";
       }
       list ssm-map {
         if-feature intf-ssm-map;
         key "ssm-map-source-addr ssm-map-group-policy";
         description "The policy for (*,G) mapping to (S,G).";

         leaf ssm-map-source-addr {
```

```
           type ssm-map-ipv4-addr-type;
           description
             "Multicast source IPv4 address.";
         }
         leaf ssm-map-group-policy {
           type string;
           description
             "Name of the policy used to define ssm-map rules.
              A device can 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-addr source-addr";
         description
           "A static multicast route, (*,G) or (S,G).
            The version of IGMP must be 3 to support (S,G).";

         leaf group-addr {
           type rt-types:ipv4-multicast-group-address;
           description
             "Multicast group IPv4 address.";
         }
         leaf source-addr {
           type rt-types:ipv4-multicast-source-address;
           description
             "Multicast source IPv4 address.";
         }
       }
     } // interface-config-attributes-igmp

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

       uses interface-common-config-attributes-mld;
       uses interface-level-config-attributes;
       leaf-list join-group {
         if-feature intf-join-group;
         type rt-types:ipv6-multicast-group-address;
         description
           "The router joins this multicast group on the interface.";
       }
       list ssm-map {
         if-feature intf-ssm-map;
         key "ssm-map-source-addr ssm-map-group-policy";
         description "The policy for (*,G) mapping to (S,G).";
```

```
         leaf ssm-map-source-addr {
           type ssm-map-ipv6-addr-type;
           description
             "Multicast source IPv6 address.";
         }
         leaf ssm-map-group-policy {
           type string;
           description
             "Name of the policy used to define ssm-map rules.
              A device can 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-addr source-addr";
         description
           "A static multicast route, (*,G) or (S,G).
            The version of MLD must be 2 to support (S,G).";

         leaf group-addr {
           type rt-types:ipv6-multicast-group-address;
           description
             "Multicast group IPv6 address.";
         }
         leaf source-addr {
           type rt-types:ipv6-multicast-source-address;
           description
             "Multicast source IPv6 address.";
         }
       }
     } // interface-config-attributes-mld

     grouping interface-state-attributes {
       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.";
           }
```

```
         }
         config false;
         mandatory true;
         description
           "Indicates whether the operational state of the interface
            is up or down.";
       }
     } // interface-state-attributes

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

       uses interface-state-attributes;
       leaf querier {
         type inet:ipv4-address;
         config false;
         mandatory true;
         description "The querier address in the subnet";
       }
       leaf-list joined-group {
         if-feature intf-join-group;
         type rt-types:ipv4-multicast-group-address;
         config false;
         description
           "The routers that joined this multicast group.";
       }
       list group {
         key "group-address";
         config false;
         description
           "Multicast group membership information
            that joined on the interface.";

         leaf group-address {
           type rt-types:ipv4-multicast-group-address;
           description
             "Multicast group address.";
         }
         uses interface-state-group-attributes;
         leaf last-reporter {
           type inet:ipv4-address;
           description
             "The IPv4 address of the last host which has sent the
              report to join the multicast group.";
         }
         list source {
           key "source-address";
           description
```

```
             "List of multicast source information
              of the multicast group.";

           leaf source-address {
             type inet:ipv4-address;
             description
               "Multicast source address in group record.";
           }
           uses interface-state-source-attributes;
           leaf last-reporter {
             type inet:ipv4-address;
             description
               "The IPv4 address of the last host which has sent the
                report to join the multicast source and group.";
           }
           list host {
             if-feature intf-explicit-tracking;
             key "host-address";
             description
               "List of hosts with the membership for the specific
                multicast source-group.";

             leaf host-address {
               type inet:ipv4-address;
               description
                 "The IPv4 address of the host.";
             }
             uses interface-state-host-attributes;
           }// list host
         } // list source
       } // list group
     } // interface-state-attributes-igmp

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

       uses interface-state-attributes;
       leaf querier {
         type inet:ipv6-address;
         config false;
         mandatory true;
         description
           "The querier address in the subnet.";
       }
       leaf-list joined-group {
         if-feature intf-join-group;
         type rt-types:ipv6-multicast-group-address;
         config false;
```

```
          description
            "The routers that joined this multicast group.";
        }
        list group {
          key "group-address";
          config false;
          description
            "Multicast group membership information
             that joined on the interface.";

          leaf group-address {
            type rt-types:ipv6-multicast-group-address;
            description
              "Multicast group address.";
          }
          uses interface-state-group-attributes;
          leaf last-reporter {
            type inet:ipv6-address;
            description
              "The IPv6 address of the last host which has sent the
               report to join the multicast group.";
          }
          list source {
            key "source-address";
            description
              "List of multicast sources of the multicast group.";

            leaf source-address {
              type inet:ipv6-address;
              description
                "Multicast source address in group record";
            }
            uses interface-state-source-attributes;
            leaf last-reporter {
              type inet:ipv6-address;
              description
                "The IPv6 address of the last host which has sent the
                 report to join the multicast source and group.";
            }
            list host {
              if-feature intf-explicit-tracking;
              key "host-address";
              description
                "List of hosts with the membership for the specific
                 multicast source-group.";

              leaf host-address {
                type inet:ipv6-address;
                description
```

```
                 "The IPv6 address of the host.";
             }
             uses interface-state-host-attributes;
           }// list host
         } // list source
       } // list group
     } // interface-state-attributes-mld

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

       leaf expire {
         type uint32;
         units seconds;
         mandatory true;
         description
           "The time left before multicast group state expires.";
       }
       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.";
           }
         }
         mandatory true;
         description
           "Filter mode for a multicast group,
            may be either include or exclude.";
       }
       leaf up-time {
         type uint32;
         units seconds;
         mandatory true;
         description
           "The elapsed time since the device created multicast group
            record.";
```

```
       }
     } // interface-state-group-attributes

     grouping interface-state-source-attributes {
       description
         "Per interface state attributes for both IGMP and MLD
          source-group records.";

       leaf expire {
         type uint32;
         units seconds;
         mandatory true;
         description
           "The time left before multicast source-group state expires.";
       }
       leaf up-time {
         type uint32;
         units seconds;
         mandatory true;
         description
           "The elapsed time since the device created multicast
            source-group record.";
       }
       leaf host-count {
         if-feature intf-explicit-tracking;
         type uint32;
         description
           "The number of host addresses.";
       }
     } // interface-state-source-attributes

     grouping interface-state-host-attributes {
       description
         "Per interface state attributes for both IGMP and MLD
          hosts of source-group records.";

       leaf host-filter-mode {
         type enumeration {
           enum "include" {
             description
               "In include mode";
           }
           enum "exclude" {
             description
               "In exclude mode.";
           }
         }
         mandatory true;
         description
```

```
           "Filter mode for a multicast membership
            host may be either include or exclude.";
       }
     } // interface-state-host-attributes

     /*
      * Configuration and Operational state data nodes (NMDA version)
      */
     augment "/rt:routing/rt:control-plane-protocols/"
       + "rt:control-plane-protocol" {
       when "derived-from-or-self(rt:type, 'igmp-mld:igmp')" {
         description
           "This augmentation is only valid for a control-plane
            protocol instance of IGMP (type 'igmp').";
       }
       description
         "IGMP augmentation to routing control plane protocol
          configuration and state.";

       container igmp {
         if-feature feature-igmp;
         description
           "IGMP configuration and operational state 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-igmp {
             if-feature interface-global-config;
             refine query-interval {
               default 125;
             }
             refine query-max-response-time {
               default 10;
             }
             refine robustness-variable {
               default 2;
             }
             refine version {
               default 2;
             }
```

```
          }
          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" {
                error-message
                  "The interface must have IPv4 configured, either "
                  + "enabled or disabled.";
              }
              description
                "Reference to an entry in the global interface list.";
            }
            uses interface-config-attributes-igmp {
              if-feature per-interface-config;
              refine last-member-query-interval {
                must "../version != 1 or "
                  + "(not(../version) and "
                  + "(../../version != 1 or not(../../version)))" {
                  error-message
                    "IGMPv1 does not support "
                    + "last-member-query-interval.";
                }
              }
              refine max-group-sources {
                must "../version = 3 or "
                  + "(not(../version) and (../../version = 3))" {
                  error-message
                    "The version of IGMP must be 3 to support the "
                    + "source specific parameters.";
                }
              }
              refine source-policy {
                must "../version = 3 or "
                  + "(not(../version) and (../../version = 3))" {
                  error-message
                    "The version of IGMP must be 3 to support the "
                    + "source specific parameters.";
                }
              }
              refine explicit-tracking {
                must "../version = 3 or "
                  + "(not(../version) and (../../version = 3))" {
                  error-message
                    "The version of IGMP must be 3 to support the "
```

```
                  + "explicit tracking function.";
              }
            }
            refine lite-exclude-filter {
              must "../version = 3 or "
                + "(not(../version) and (../../version = 3))" {
                error-message
                  "The version of IGMP must be 3 to support the "
                  + "simplified EXCLUDE filter in the Lightweight "
                  + "IGMPv3 protocol.";
              }
            }
          }
          uses interface-state-attributes-igmp;
        } // interface
      } // interfaces

      /*
       * Actions
       */
      action clear-groups {
        if-feature action-clear-groups;
        description
          "Clears the specified IGMP cache entries.";

        input {
          choice interface {
            mandatory true;
            description
              "Indicates the interface(s) from which the cache
               entries are cleared.";
            case name {
              leaf interface-name {
                type leafref {
                  path "/rt:routing/rt:control-plane-protocols/"
                    + "rt:control-plane-protocol/"
                    + "igmp-mld:igmp/igmp-mld:interfaces/"
                    + "igmp-mld:interface/igmp-mld:interface-name";
                }
                description
                  "Name of the IGMP interface.";
              }
            }
            case all {
              leaf all-interfaces {
                type empty;
                description
                  "IGMP groups from all interfaces are cleared.";
              }
```

```
              }
            }
            leaf group-address {
              type union {
                type enumeration {
                  enum '*' {
                    description
                      "Any group address.";
                  }
                }
                type rt-types:ipv4-multicast-group-address;
              }
              mandatory true;
              description
                "Multicast group IPv4 address.
                 If the value '*' is specified, all IGMP group entries
                 are cleared.";
            }
            leaf source-address {
              type rt-types:ipv4-multicast-source-address;
              mandatory true;
              description
                "Multicast source IPv4 address.
                 If the value '*' is specified, all IGMP source-group
                 entries are cleared.";
            }
          }
        } // action clear-groups
      } // igmp
    } //augment

    augment "/rt:routing/rt:control-plane-protocols/"
      + "rt:control-plane-protocol" {
      when "derived-from-or-self(rt:type, 'igmp-mld:mld')" {
        description
          "This augmentation is only valid for a control-plane
           protocol instance of IGMP (type 'mld').";
      }
      description
        "MLD augmentation to routing control plane protocol
         configuration and state.";

      container mld {
        if-feature feature-mld;
        description
          "MLD configuration and operational state 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-mld {
             if-feature interface-global-config;
             refine last-member-query-interval {
               default 1;
             }
             refine query-interval {
               default 125;
             }
             refine query-max-response-time {
               default 10;
             }
             refine require-router-alert {
               default true;
             }
             refine robustness-variable {
               default 2;
             }
             refine version {
               default 2;
             }
           }
           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" {
                 error-message
                   "The interface must have IPv6 configured, either "
                   + "enabled or disabled.";
               }
               description
                 "Reference to an entry in the global interface list.";
             }
             uses interface-config-attributes-mld {
               if-feature per-interface-config;
               refine max-group-sources {
```

```
                must "../version = 2 or "
                  + "(not(../version) and "
                  + "(../../version = 2 or not(../../version)))" {
                  error-message
                    "The version of MLD must be 2 to support the "
                    + "source specific parameters.";
                }
              }
              refine source-policy {
                must "../version = 2 or "
                  + "(not(../version) and "
                  + "(../../version = 2 or not(../../version)))" {
                  error-message
                    "The version of MLD must be 2 to support the "
                    + "source specific parameters.";
                }
              }
              refine explicit-tracking {
                must "../version = 2 or "
                  + "(not(../version) and "
                  + "(../../version = 2 or not(../../version)))" {
                  error-message
                    "The version of MLD must be 2 to support the "
                    + "explicit tracking function.";
                }
              }
              refine lite-exclude-filter {
                must "../version = 2 or "
                  + "(not(../version) and "
                  + "(../../version = 2 or not(../../version)))" {
                  error-message
                    "The version of MLD must be 2 to support the "
                    + "simplified EXCLUDE filter in the Lightweight "
                    + "MLDv2 protocol.";
                }
              }
            }
            uses interface-state-attributes-mld;
          } // interface
        } // interfaces

        /*
         * Actions
         */
        action clear-groups {
          if-feature action-clear-groups;
          description
            "Clears the specified MLD cache entries.";
```

```
           input {
             choice interface {
               mandatory true;
               description
                 "Indicates the interface(s) from which the cache
                  entries are cleared.";
               case name {
                 leaf interface-name {
                   type leafref {
                     path "/rt:routing/rt:control-plane-protocols/"
                       + "rt:control-plane-protocol/"
                       + "igmp-mld:mld/igmp-mld:interfaces/"
                       + "igmp-mld:interface/igmp-mld:interface-name";
                   }
                   description
                     "Name of the MLD interface.";
                 }
               }
               case all {
                 leaf all-interfaces {
                   type empty;
                   description
                     "MLD groups from all interfaces are cleared.";
                 }
               }
             }
             leaf group-address {
               type union {
                 type enumeration {
                   enum '*' {
                     description
                       "Any group address.";
                   }
                 }
                 type rt-types:ipv6-multicast-group-address;
               }
               description
                 "Multicast group IPv6 address.
                  If the value '*' is specified, all MLD group entries
                  are cleared.";
             }
             leaf source-address {
               type rt-types:ipv6-multicast-source-address;
               description
                 "Multicast source IPv6 address.
                  If the value '*' is specified, all MLD source-group
                  entries are cleared.";
             }
           }
```

```
       } // action clear-mld-groups
     } // mld
   } // augment
}
<CODE ENDS>
```

5. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

Under /rt:routing/rt:control-plane-protocols /rt:control-plane-protocol/igmp-mld:igmp,

igmp-mld:global

This subtree specifies the configuration for the IGMP attributes at the global level on an IGMP instance. Modifying the configuration can cause IGMP membership to be deleted or reconstructed on all the interfaces of an IGMP instance.

igmp-mld:interfaces

This subtree specifies the configuration for the IGMP attributes at the interface-global level on a IGMP instance. Modifying the configuration can cause IGMP membership to be deleted or reconstructed on all the interfaces of an IGMP instance.

igmp-mld:interfaces/interface

This subtree specifies the configuration for the IGMP attributes at the interface level on an IGMP instance. Modifying the configuration can cause IGMP membership to be deleted or reconstructed on a specific interface of an IGMP instance.

Under /rt:routing/rt:control-plane-protocols /rt:control-plane-protocol/igmp-mld:mld,

igmp-mld:global

This subtree specifies the configuration for the MLD attributes at the global level on an MLD instance. Modifying the configuration can cause MLD membership to be deleted or reconstructed on all the interfaces of an MLD instance.

igmp-mld:interfaces

This subtree specifies the configuration for the MLD attributes at the interface-global level on an MLD instance. Modifying the configuration can cause MLD membership to be deleted or reconstructed on all the interfaces of an MLD instance.

igmp-mld:interfaces/interface

This subtree specifies the configuration for the MLD attributes at the interface level on a device. Modifying the configuration can cause MLD membership to be deleted or reconstructed on a specific interface of an MLD instance.

Unauthorized access to any data node of these subtrees can adversely affect the membership records of multicast routing subsystem on the local device. This may lead to network malfunctions, delivery of packets to inappropriate destinations, and other problems.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/rt:routing/rt:control-plane-protocols
/rt:control-plane-protocol/igmmp-mld:igmp

/rt:routing/rt:control-plane-protocols
/rt:control-plane-protocol/igmp-mld:mld

Unauthorized access to any data node of the above subtree can disclose the operational state information of IGMP or MLD on this device.

Some of the action operations in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control access to these operations. These are the operations and their sensitivity/vulnerability:

/rt:routing/rt:control-plane-protocols
/rt:control-plane-protocol/igmmp-mld:igmp/igmmp-mld:clear-groups

/rt:routing/rt:control-plane-protocols
/rt:control-plane-protocol/igmp-mld:mld/igmp-mld:clear-groups

Unauthorized access to any of the above action operations can delete the IGMP or MLD membership records on this device.

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 [RFC6020]:

----

name: ietf-igmp-mld

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

prefix: igmp-mld

reference: RFC XXXX

---

7. Acknowledgments

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

8. Contributing Authors

Yisong Liu
Huawei Technologies
Huawei Bldg., No.156 Beiqing Rd.
Beijing 100095
China

Email: liuyisong@huawei.com

9. References

9.1. Normative References

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.

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

[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.

[RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.

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

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

[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.

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, July 2013.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.

[RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, December 2017.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, March 2018.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, March 2018.

[RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, March 2018.

[RFC8344] M. Bjorklund, "A YANG Data Model for IP Management", RFC8344, March 2018.

[RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, March 2018.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.

[RFC8519] M. Jethanandani, S. Agarwal, L. Huang and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, March 2019.

9.2. Informative References

[RFC3569] Bhattacharyya, S., Ed., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, July 2003.

[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.

[RFC5790] H. Liu, W. Cao and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, February 2010.

[RFC6636] H. Asaeda, H. Liu and Q. Wu, "Tuning the Behavior of the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) for Routers in Mobile and Wireless Networks", RFC 6636, May 2012.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, March 2018.

[RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", RFC 8407, October 2018.

[I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher's Event Streams", draft-ietf-netconf-subscribed-notifications-26 (work in progress), May 2019.

[I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-25 (work in progress), May 2019.

Authors' Addresses

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

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

Email: guofeng@huawei.com

Mahesh Sivakumar
Juniper Networks
1133 Innovation Way
Sunnyvale, California
USA

Email: sivakumar.mahesh@gmail.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 Working Group LM. Contreras
Internet-Draft Telefonica
Intended status: Informational CJ. Bernardos
Expires: May 13, 2019 Universidad Carlos III de Madrid
H. Asaeda
NICT
N. Leymann
Deutsche Telekom
November 9, 2018

Requirements for the extension of the IGMP/MLD proxy functionality to support multiple upstream interfaces
draft-ietf-pim-multiple-upstreams-reqs-08

Abstract

The purpose of this document is to define the requirements for a MLD (for IPv6) or IGMP (for IPv4) proxy with multiple interfaces covering a variety of applicability scenarios. The referred scenarios, while describing not sophisticated service situations, present cases that existing technology does not allow to solve in a simplistic manner. This document is then intended to serve as input for future documents defining the support of multiple upstream interfaces by IGMP/MLD proxies being compliant with the aforementioned requirements.

Table of Contents

1. Introduction

The aim of this document is to define the functionality that an IGMP/MLD proxy with multiple upstream interfaces should have in order to support different scenarios of applicability in both fixed and mobile networks. IGMP/MLD proxis are a generic solution very much deployed in existing carrier networks. An extension to them in the sense of supporting multiple upstream interfaces can provide a more flexible and lightweight solution than other potential alternatives that could face more complexities (like multi-domain routing in the case of PIM, or the need of some external elements -e.g., controllers- if the coordination of actions required lays outside the proxy).

The functional behavior of an IGMP/MLD proxy with multiple upstream interfaces here described is needed in order to simplify node functionality and to ensure an easier deployment of multicast capabilities in all the use cases described in this document.

For doing that, a number of scenarios are described, representing current deployments and needs from operator's networks. From that scenarios, certain requirements are identified as needed to simplify operational situations, enable optimized service delivery, etc. Those represent functional requirements to be satisfied by IGMP/MLD proxies with multiple upstream interfaces. These functional requirements reflect the need of coordinating actions from a single element in the network (i.e., the IGMP/MLD proxy), optimizing the delivery of the content within the network at any time.

Any Source Multicast (ASM) [RFC1112] and Source-Specific Multicast (SSM) [RFC4607] represent different service models at the time of subscribing to multicast groups by means of IGMPv3 [RFC3376], [RFC5790] and MLDv2 [RFC3810]. When using ASM a receiver joins a group indicating only the desired group address to be received. In the case of SSM, a receiver indicates the specific source address as well as a group address from where the multicast content is received. Both service models are taken into account along this document, and the specific requirements are derived from them.

2. Terminology

This document uses the terminology defined in [RFC4605]. Specifically, the definition of Upstream and Downstream interfaces, which are repeated here for completeness.

Upstream interface: A proxy device's interface in the direction of the root of the tree. Also called the "Host interface".

Downstream interface: Each of a proxy device's interfaces that is not in the direction of the root of the tree. Also called the "Router interfaces".

3. Problem statement

The concept of IGMP/MLD proxy with several upstream interfaces has emerged as a way of optimizing (and in some cases enabling) service delivery scenarios where separate multicast service providers are reachable through the same access network infrastructure. Figure 1 presents the conceptual model under consideration.

```
            downstream        upstream
            interface       interface A
                 |               |
                 |               |     _______________
                 |   +-------+   v    /               \
                 |   |       O-------( Multicast Set 1 )
   +----------+  v   | IGMP/ |        \_______________/
   | Listener |------|  MLD  |         _______________
   +----------+      | Proxy |        /               \
                     |       O-------( Multicast Set 2 )
                     +-------+   ^    \_______________/
                                 |
                                 |
                              upstream
                            interface B
```

Figure 1: Concept of IGMP/MLD proxy with multiple upstream interfaces

This document is focused on both fixed and mobile network scenarios. Applicability of IGMP/MLD proxies with multiple upstream interfaces in mobile environments has been previously identified as beneficial in scenarios as the ones described in [RFC6224] and [RFC7287].

In the case of fixed networks, multicast wholesale services in a competitive residential market require an efficient distribution of multicast traffic from different operators or content providers, i.e. the incumbent operator and a number of alternative providers, on the network infrastructure of the former. Existing proposals are based on the use of PIM routing from the metro/core network, and multicast traffic aggregation on the same tree. A different approach could be achieved with the use of an IGMP/MLD proxy with multiple upstream interfaces, each of them pointing to a distinct multicast router in the metro/core border which is part of separated multicast trees deep in the network. Figure 2 graphically describes this scenario.

```
                      downstream        upstream
                       interface       interface A
                          |                |
                          |                |     _______________
                          |   +--------+   v    /               \
                          |   |  Aggr. O-------( Multicast Set 1 )
                          |   | Switch |        \_______________/
                 +----+   v   |        |     (e.g. from the Incumbent
                 | AN |-------| (IGMP  |             Operator)
                 +----+       |  /MLD  |         _______________
                 (e.g.        | Proxy) |        /               \
                 DSLAM        |        O-------( Multicast Set 2 )
                 /OLT)        +--------+   ^    \_______________/
                                           | (e.g. from an Alternative
                                           |          Provider)
                                           |
                                        upstream
                                       interface B
```

Figure 2: Example of usage of an IGMP/MLD proxy with multiple upstream interfaces in a fixed network scenario

Since those scenarios can motivate distinct needs in terms of IGMP/ MLD proxy functionality, it is necessary to consider a comprehensive approach, looking at the possible scenarios, and establishing a minimum set of requirements which can allow the operation of a versatile IGMP/MLD proxy with multiple upstream interfaces as a common entity to all of them (i.e., no different kinds of proxies depending on the scenario, but a common proxy applicable to all the potential scenarios).

4. Scenarios of applicability

Having multiple upstream interfaces creates a new decision space for delivering the proper multicast content to the subscriber. Basically it is now possible to implement channel-based (i.e., leveraging on multicast group IP address) or subscriber-based (i.e., referenced to the subscriber IP address) upstream selection, according to mechanisms or policies that could be defined for the multicast service provisioning.

This section describes in detail a number of scenarios of applicability of an IGMP/MLD proxy with multiple upstream interfaces in place. A number of requirements for the IGMP/MLD proxy functionality are identified from those scenarios.

All the exemplary scenarios here described are based on the support of two upstream interfaces. However, all of them are applicable also to the support of more than two upstream interfaces.

4.1. Fixed network scenarios

Residential broadband users get access to multiple IP services through fixed network infrastructures. End user's equipment is connected to an access node, and the traffic of a number of access nodes is collected in aggregation switches.

For the multicast service, the use of an IGMP/MLD proxy with multiple upstream interfaces in those switches appears as a simple and straightforward solution.

4.1.1. Multicast wholesale offer for residential services

This scenario has been already introduced in the previous section, and can be seen in Figure 2. There are two different operators, the one operating the fixed network where the end user is connected (e.g., typically an incumbent operator), and the one providing the Internet service to the end user (e.g., an alternative Internet service provider). Both can offer multicast streams that can be subscribed by the end user, independently of which provider contributes with the content.

Note that it is assumed that both providers offer distinct multicast groups. However, more than one subscription to multicast channels of different providers could take place simultaneously.

4.1.1.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding provider's multicast router.
- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the providers to the corresponding end user.
- o The IGMP/MLD proxy should be able to support ASM and SSM at the time of requesting the content. Since the use case assumes that each provider offers distinct multicast groups, the IGMP/MLD proxy should be able to identify inconsistencies in the SSM requests, that is, the case in which for an (S, G) request the source S does not deliver a the group G.

4.1.2. Multicast resiliency

In current PIM-based solutions [RFC7063], the resiliency of the multicast distribution relays on the routing capabilities provided by protocols like PIM [RFC7761] and VRRP [RFC5798]. A simpler scheme could be achieved by implementing different upstream interfaces on IGMP/MLD proxies, providing path diversity through the connection to distinct leaves of a given multicast tree.

It is assumed that only one of the upstream interfaces is active in receiving the multicast content, while the other is up and in standby mode for fast switching. The objective is to avoid video delivery affection that could imply play out interruption or buffering on the user side. Service parameters like the ones defined in [Y.1540] (such as packet loss ratio) or in [RFC4445] (like the delay factor) can be considered as parameters to be assesed from the service perspective. For instance, [TECH.3361-1] could be considered as a SLA framework to be satisfied in this case.

4.1.2.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages received in the active upstream to the end users, while ignoring the control messages of the standby upstream interface.

- o The IGMP/MLD proxy should be able of rapidly switching from the active to the standby upstream interface in case of network failure, transparently to the end user.

- o The IGMP/MLD proxy should be able to deliver IGMP/MLD messages sent by the end user (for both ASM and SSM modes) to the corresponding active upstream interface.

4.1.3. Load balancing for multicast traffic in the metro segment

A single upstream interface in existing IGMP/MLD proxy functionality [RFC4605] typically forces the distribution of all the channels on the same path in the last segment of the network. The metro and backhaul network is usually built using ring topologies. The devices in the ring implement IGMP/MLD functionality to join the content. Multiple upstream interfaces could naturally help to split the content demand, alleviating the bandwidth requirements in the overall metro segment by allowing some of the channels to follow the protection path, where spare capacity is vacant under normal conditions. This will allow, for instance, to absorb traffic peaks when a high number of channels (more than the expected on average) is requested.

4.1.3.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding multicast router which provides the channel of interest.

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the multicast routers to the corresponding end user.

- o The IGMP/MLD proxy should be able to decide which upstream interface is selected for any new channel request according to defined criteria (e.g., load balancing).

- o In the case of ASM, the IGMP/MLD proxy should be able to balance the traffic as a function of the group G requested. In the case of SSM, the load balancing mechanism could also consider the source S for the decision. In any case, the criteria will follow the olicies defined by the network operator. Such policies can be influenced by the user requesting the service, for instance through the subscription to some channels being offered by a third party (which has reached an agreement with the provider for delivering that content in its network).

4.1.4. Network merging with different multicast services

In some network merging situations, the multicast services provided before in each of the merged networks are maintained for the respective customer base (usually in a temporal fashion until the multicast service is redefined in a new single offer, but not necessarily, or not in short term, e.g. because of commercial agreements for each of the previous service offers).

In order to assist that network merging situations, IGMP/MLD proxies with multiple upstream interfaces can help in the transition simplifying the service provisioning and facilitating service continuity.

4.1.4.1. Requirements

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding multicast router which provides the channel of interest, according to the service subscription.

- o The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the multicast routers to the corresponding end user, according to the service subscription.

- The IGMP/MLD proxy should be able to decide which upstream interface is selected for any new channel request according to defined criteria (e.g., service subscription).

- For this use case, the usage of SSM can simplify the decision of the IGMP/MLD proxy. For ASM the decision should be assisted by further information like the service to which the end user is subscribed (e.g., taking into account what is the original network from where the end user was part previous to the network merge situation).

4.1.5. Multicast service migration

This scenario considers the situation where a multicast service needs to be migrated from one upstream interface to another upstream interface (e.g. because of changes inside the service provider's network). The migration should be "smooth" and without any service interruption. In this case the multicast content is initially offered in both upstream interfaces and the proxy dynamically switches from the first to the second upstream interface, according to certain policies, and enabling to shut down the first upstream interface once the migration is completed.

4.1.5.1. Requirements

- The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding multicast router before and after the service migration.

- The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the multicast routers to the corresponding end user, according to the situation of the user with respect to the service migration.

- The IGMP/MLD proxy should be able to decide which upstream interface corresponds to each user, according to the situation of the user with respect to the service migration, i.e., the status of the user with respect the platform migration as purely operational situation while transitioning from one platform to another in a smooth manner.

- The IGMP/MLD proxy should be able to decide which upstream interface corresponds to each ASM or SSM request, according to the situation of the group and source included in the request with respect to the service migration.

4.2. Mobile network scenarios

Mobile networks offer different alternatives for multicast distribution.

One of them is defined by 3GPP [TS23.246] for the Multimedia Broadcast Multicast Service (MBMS). In this case, a MBMS gateway (MBMS GW) is connected to multiple evolved Node B (eNodeB) -- which are the base stations connecting the mobile handsets with the network wirelessly [TS36.300] -- for data distribution by means of IP multicast. The MBMS GW delivers the IP multicast groups. The eNodeB joins the appropriate group multicast address allocated by the MBMS GW to receive the content data. At this distribution level, an IGMP/MLD proxy could be part of the transport infrastructure providing connectivity to several distributed eNodeBs. The potential scenarios from this case do not essentially differentiate from the ones described for the fixed network scenarios, so the same situations and requirements apply.

Another alternative is given by Proxy Mobile IPv6 (PMIPv6) protocol for IP mobility management [RFC5213]. PMIPv6 is one of the mechanisms adopted by the 3GPP to support the mobility management of non-3GPP terminals in future Evolved Packet System (EPS) networks. PMIPv6 allows a Media Access Gateway (MAG) to establish a distinct bi-directional tunnel with different Local Mobility Anchors (LMAs), being each tunnel shared by the attached Mobile Nodes (MNs). Each mobile node is associated with a corresponding LMA, which keeps track of its current location, that is, the MAG where the mobile node is attached. As the basic solution for the distribution of multicast traffic within a PMIPv6 domain, [RFC6224] makes use of the bi-directional LMA-MAG tunnels. The use of an MLD proxy supporting multiple upstream interfaces can improve the performance and the scalability of multicast-capable PMIPv6 domains, for both multicast listener and multicast source mobility. Once again, the potential scenarios in this case are contained into the ones described for the fixed network scenarios, so the same situations and requirements apply.

5. Summary of requirements

Following the analysis above, a number of different requirements can be identified by the IGMP/MLD proxy to support multiple upstream interfaces. The following table summarizes these requirements.

Functionality	Multicast Wholesale	Multicast Resiliency	Load Balancing	Network Merging	Network Migration
Upstream Control Delivery	X	X	X	X	X
Downstr. Control Delivery	X	X	X	X	X
Active / Standby Upstream		X			
Upstr i/f selection per group			X	X	
Upstr i/f selection all group		X			X
ASM	X	X	X	X	X
SSM	X	X	X		X

Figure 3: Functionality needed on IGMP/MLD proxy with multiple upstream interfaces per application scenario

6. Security Considerations

All the security considerations in [RFC4605] are directly applicable to this proposal.

7. IANA Considerations

There are no IANA considerations.

8. Acknowledgements

The authors would like to thank (in alphabetical order) Alvaro Retana, Thomas C. Schmidt, Stig Venaas and Dirk von Hugo for their comments and suggestions.

9. References

9.1. Normative References

[RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, DOI 10.17487/RFC1112, August 1989, <https://www.rfc-editor.org/info/rfc1112>.

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <https://www.rfc-editor.org/info/rfc4605>.

[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>.

[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>.

9.2. Informative References

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <https://www.rfc-editor.org/info/rfc3376>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <https://www.rfc-editor.org/info/rfc3810>.

[RFC4445] Welch, J. and J. Clark, "A Proposed Media Delivery Index (MDI)", RFC 4445, DOI 10.17487/RFC4445, April 2006, <https://www.rfc-editor.org/info/rfc4445>.

[RFC5213] Gundavelli, S., Ed., Leung, K., Devarapalli, V., Chowdhury, K., and B. Patil, "Proxy Mobile IPv6", RFC 5213, DOI 10.17487/RFC5213, August 2008, <https://www.rfc-editor.org/info/rfc5213>.

[RFC5790] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, DOI 10.17487/RFC5790, February 2010, <https://www.rfc-editor.org/info/rfc5790>.

[RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <https://www.rfc-editor.org/info/rfc5798>.

[RFC6224] Schmidt, T., Waehlisch, M., and S. Krishnan, "Base Deployment for Multicast Listener Support in Proxy Mobile IPv6 (PMIPv6) Domains", RFC 6224, DOI 10.17487/RFC6224, April 2011, <https://www.rfc-editor.org/info/rfc6224>.

[RFC7063] Zheng, L., Zhang, J., and R. Parekh, "Survey Report on Protocol Independent Multicast - Sparse Mode (PIM-SM) Implementations and Deployments", RFC 7063, DOI 10.17487/RFC7063, December 2013, <https://www.rfc-editor.org/info/rfc7063>.

[RFC7287] Schmidt, T., Ed., Gao, S., Zhang, H., and M. Waehlisch, "Mobile Multicast Sender Support in Proxy Mobile IPv6 (PMIPv6) Domains", RFC 7287, DOI 10.17487/RFC7287, June 2014, <https://www.rfc-editor.org/info/rfc7287>.

[TECH.3361-1]
European Broadcasting Union, "Service Level Agreement for media transport services", EBU TECH.3361-1, September 2014.

[TS23.246]
"TS 23.246 Multimedia Broadcast/Multicast Service (MBMS); Architecture and functional description (Release 14) V14.1.0.", 3GPP TS 23.246 V14.1.0 , December 2016.

[TS36.300]
3GPP, "Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2", 3GPP TS 36.300 10.11.0, September 2013.

[Y.1540] ITU-T, "Internet protocol data communication service - IP packet transfer and availability performance parameters", ITU-T Y.1540, July 2016.

Authors' Addresses

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://lmcontreras.com/

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

Hitoshi Asaeda
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: asaeda@nict.go.jp

Nic Leymann
Deutsche Telekom
Germany

Email: n.leymann@telekom.de

Network Working Group S. Venaas
Internet-Draft IJ. Wijnands
Intended status: Experimental M. Mishra
Expires: April 25, 2019 Cisco Systems, Inc.
M. Sivakumar
Juniper Networks
October 22, 2018

PIM Flooding Mechanism and Source Discovery for BIER
draft-venaas-bier-pfm-sd-00

Abstract

PIM Flooding Mechanism and Source Discovery (PFM-SD) is a mechanism for source discovery within a PIM domain. PIM signaling over BIER has been defined, allowing for BIER to interoperate with PIM. This document defines PFM-SD over BIER, such that PFM-SD can be used by PIM in a PIM domain to discover sources that are reachable via BIER. Also, this document provides PFM-SD extensions to discover the BIER ingress router closest to the source. This can be used by BIER overlays, such as PIM signaling over BIER, to determine which router to signal.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

PIM Flooding Mechanism (PFM) and Source Discovery (SD) [RFC8364] provides a generic flooding mechanism for distributing information throughout a PIM domain. In particular it allows for source discovery. There are various deployment scenarios where PIM and BIER need to co-exist. For instance, consider migration scenarios where a few routers in a PIM domain are upgraded to support BIER. In that case, one may use PIM Signaling Through BIER Core [I-D.ietf-bier-pim-signaling], allowing PIM to build trees passing through the BIER routers. This document defines PFM over BIER. This allows PFM to pass through the BIER routers, allowing PFM to be used in the PIM domain.

One challenge with PIM signaling over BIER [I-D.ietf-bier-pim-signaling] is to determine which BIER router is closest to the source. A number of options are discussed in that document. This document provides an alternative solution for discovering which BIER router to signal. It may also be used with other signaling mechanisms such as IGMP/MLD [I-D.ietf-bier-mld]. This is achieved by introducing two new PFM TLVs. When a BIER router forwards a PFM message into BIER, it adds a new TLV specifying the BIER sub-domain, its BFR-ID and its BIER prefix. Also, any Group Source Holdtime TLVs, defined in [RFC8364], are replaced with new TLVs that include the router's cost of reaching the sources.

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].

2. PFM over BIER

When a BIER enabled router accepts a PFM message from a PIM neighbor according to [RFC8364], it SHOULD in addition to the forwarding defined in [RFC8364], also send a copy to all BIER routers (an implementation SHOULD allow the set of BIER routers to send PFM messages to, to be configured).

When a router receives a BIER encapsulated PFM message, it MUST process the message according to [RFC8364], except there is no requirement for the message to come from a PIM neighbor, and there is no RPF check. The message MUST be forwarded out on the PIM interfaces according to [RFC8364]. It MAY also be BIER forwarded, if the router acts as a border router between BIER domains.

3. PFM Ingress BIER Router TLV

When a router is forwarding a PFM message into a BIER domain, it MUST add this TLV. If the TLV is already present, all occurrences should be removed. This TLV encodes the BIER prefix, sub-domain ID and BFR-ID of the router. This TLV SHOULD only be present within the BIER domain. When a router receives a PFM message with this TLV, all occurrences of the TLV SHOULD be removed. If the router is forwarding the message into a new BIER domain, it should add a new TLV with its own prefix, sub-domain ID and BFR-ID. A PFM message is expected to have at most one such TLV. A router MUST NOT add more than one such TLV. When forwarding a PFM message, the TLV in the received message MUST be removed from the forwarded message.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|         Type = TBD          |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Sub-domain-id |   Reserved    |            BFR-id             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   BFR-prefix (4 or 16 octets)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

0: The Transitive bit is set to 0.

Type: Type is TBD.

Length: The length of the value in octets.

Sub-domain-id: The ID of the sub-domain that this PFM is forwarded into. The length is 1 octet.

Reserved: MUST be set to 0, and ignored when received. The length is 1 octet.

BFR-id: The BFR-id of the router that added this TLV in the sub-domain specified. The length is 2 octets.

BFR-prefix: The BFR-prefix of the router that added this TLV in the sub-domain specified. This length is 4 octets for IPv4 and 16 octets for IPv6.

4. Group Source Holdtime Metric TLV

When a router forwards a PFM message into a BIER domain, it should replace all Group Source Holdtime TLVs defined in [RFC8364] with the Group Source Holdtime Metric TLVs defined here. They are the same, except here we also add metric preference and metric. The metric preference and metric MUST be set to this router's metric and preference to reach the specified source. If the source is not reachable, the TLV MUST be omitted. This TLV is used together with the PFM Ingress BIER Router TLV is used to indicate the ingress router's cost of reaching the source.

When a router receives a message containing this TLV, it SHOULD store this information, but it MUST NOT forward these TLVs. If forwarding into another BIER domain, the metric preference and metric MUST be updated with this router's cost of reaching the source. If forwarding into a PIM domain, all the TLVs SHOULD be replaced with Group Source Holdtime TLVs as defined in [RFC8364]. The same information is used, except that the metric preference and metric are left out. One could potentially make use of the metric in a PIM domain as well, but it is not clear whether this is useful, and the PIM routers may not support this TLV.

```
     0                   1                   2                   3
     0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|         Type = TBD          |             Length            |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           Group Address (Encoded-Group format)                |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Src Count            |          Src Holdtime         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Src Address 1 (Encoded-Unicast format)               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       Metric Preference 1                     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                            Metric 1                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Src Address 2 (Encoded-Unicast format)               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       Metric Preference 2                     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                            Metric 2                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                               .                               |
    |                               .                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |          Src Address m (Encoded-Unicast format)               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                       Metric Preference m                     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                            Metric m                           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

0: The Transitive bit is set to 0.

Type: Type is TBD.

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].

Metric Preference: Preference value assigned to the unicast routing protocol that provided the route to the source.

Metric: The unicast routing table metric associated with the route used to reach the source. The metric is in units applicable to the unicast routing protocol used.

5. BIER signaling enhancements

A BIER border router SHOULD cache all the Group Source Holdtime Metric TLVs it receives, along with the respective PFM Ingress BIER Router TLV. This allows the router to determine which sources are active, and which BIER border router is closest to the source. The sub-domain ID, BFR-id and BFR-prefix in the TLV provide the necessary information for use by signaling mechanisms such as [I-D.ietf-bier-pim-signaling] to signal the preferred ingress router. It may also be used by [I-D.ietf-bier-mld]. IGMP/MLD reports would generally be sent to all BIER routers as it is not known which sources are active and which routers can reach them. But by using the enhancements in this document, a source-specific report can be sent to the router closest to the source. Also a group report might be set to the set of routers that are closest to the sources for that group. This reduces the amount of receiver state on the BIER routers, and also the amount of messages each routers needs to process.

6. Security Considerations

TBD

7. IANA Considerations

This document defines two new PFM TLVs that needs to be assigned from the "PIM Flooding Mechanism Message Types" registry.

8. References

8.1. Normative References

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

[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>.

[RFC8364] Wijnands, IJ., Venaas, S., Brig, M., and A. Jonasson, "PIM Flooding Mechanism (PFM) and Source Discovery (SD)", RFC 8364, DOI 10.17487/RFC8364, March 2018, <https://www.rfc-editor.org/info/rfc8364>.

8.2. Informative References

[I-D.ietf-bier-mld] Pfister, P., Wijnands, I., Venaas, S., Wang, C., Zhang, Z., and M. Stenberg, "BIER Ingress Multicast Flow Overlay using Multicast Listener Discovery Protocols", draft-ietf-bier-mld-01 (work in progress), June 2018.

[I-D.ietf-bier-pim-signaling] Bidgoli, H., Dolganow, A., Kotalwar, J., Xu, F., mishra, m., and Z. Zhang, "PIM Signaling Through BIER Core", draft-ietf-bier-pim-signaling-04 (work in progress), October 2018.

Authors' Addresses

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

Email: stig@cisco.com

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

Email: ice@cisco.com

Mankamana Mishra
Cisco Systems, Inc.
Tasman Drive
San Jose CA 95134
USA

Email: mankamis@cisco.com

Mahesh Sivakumar
Juniper Networks
1133 Innovation Way
Sunnyvale CA 94089
USA

Email: sivakumar.mahesh@gmail.com

PIM Working Group H. Zhao
Internet Draft Ericsson
Intended status: Standards Track X. Liu
Expires: January 02, 2020 Volta
Y. Liu
Huawei
M. Panchanathan
Cisco
M. Sivakumar
Juniper

July 03, 2019

A Yang Data Model for IGMP/MLD Proxy
draft-zhao-pim-igmp-mld-proxy-yang-03.txt

Abstract

This document defines a YANG data model that can be used to configure and manage Internet Group Management Protocol (IGMP) or Multicast Listener Discovery (MLD) proxy devices. The YANG module in this document conforms to Network Management Datastore Architecture (NMDA).

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

This Internet-Draft will expire on January 02, 2020.

Copyright Notice

Table of Contents

1. Introduction

This document defines a YANG [RFC6020] data model for the management of Internet Group Management Protocol (IGMP) or Multicast Listener Discovery (MLD) proxy devices.

The YANG module in this document conforms to the Network Management Datastore Architecture defined in [RFC8342]. The "Network Management Datastore Architecture" (NMDA) adds the ability to inspect the current operational values for configuration, allowing clients to use identical paths for retrieving the configured values and the operational values.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119].

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

1.2. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is as follows:

- o Brackets "[" and "]" enclose list keys.

- o Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).

- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.

- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").

- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Design of Data Model

The model covers Considerations for Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD) - Based Multicast Forwarding ("IGMP/MLD Proxying") [RFC4605].

The goal of this document is to define a data model that provides a common user interface to IGMP/MLD proxy. This document provides freedom for vendors to adapt this data model to their product implementations.

2.1. Overview

The IGMP/MLD proxy YANG module defined in this document has all the common building blocks for the IGMP/MLD proxy protocol.

The YANG module augments /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol to enable IGMP/MLD proxy and configure other related parameters.

This YANG module follows the Guidelines for YANG Module Authors (NMDA) [draft-dsdt-nmda-guidelines-01]. This NMDA ("Network Management Datastore Architecture") architecture provides an architectural framework for datastores as they are used by network management protocols such as NETCONF [RFC6241], RESTCONF [RFC8040] and the YANG [RFC7950] data modeling language.

2.2. Augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol

The YANG module augments /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol to enable IGMP/MLD proxy under the upstream interface. There is also a constraint to make sure the upstream interface for IGMP/MLD proxy should not be configured PIM.

```
module: ietf-igmp-mld-proxy
  augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
    +--rw igmp-proxy {feature-igmp-proxy}?
       +--rw interfaces
          +--rw interface* [interface-name]
             +--rw interface-name    if:interface-ref
             +--rw version?          uint8
             +--rw enable?           boolean
             +--ro group* [group-address]
                +--ro group-address    inet:ipv4-address
                +--ro up-time?         uint32
                +--ro filter-mode?     enumeration
                +--ro source* [source-address]
                   +--ro source-address          inet:ipv4-address
                   +--ro up-time?                uint32
                   +--ro filter-mode?            enumeration
                   +--ro downstream-interface* [interface-name]
                      +--ro interface-name    if:interface-ref
                      +--ro filter-mode?      enumeration
```

```
  augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
    +--rw mld-proxy {feature-mld-proxy}?
       +--rw interfaces
          +--rw interface* [interface-name]
             +--rw interface-name    if:interface-ref
             +--rw version?          uint8
             +--rw enable?           boolean
             +--ro group* [group-address]
                +--ro group-address    inet:ipv6-address
                +--ro up-time?         uint32
                +--ro filter-mode?     enumeration
                +--ro source* [source-address]
                   +--ro source-address          inet:ipv6-address
                   +--ro up-time?                uint32
                   +--ro filter-mode?            enumeration
                   +--ro downstream-interface* [interface-name]
                      +--ro interface-name    if:interface-ref
                      +--ro filter-mode?      enumeration
```

3. IGMP/MLD Proxy YANG Module

```
<CODE BEGINS> file ietf-igmp-mld-proxy@2019-07-03.yang
module ietf-igmp-mld-proxy {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld-proxy";
  // replace with IANA namespace when assigned
  prefix imp;

  import ietf-inet-types {
    prefix inet;
  }

  import ietf-interfaces {
    prefix if;
  }

  import ietf-routing {
    prefix rt;
  }

  import ietf-pim-base {
    prefix pim-base;
  }

  organization
    "IETF PIM Working Group";
```

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

     Editors:  Hongji Zhao
               <mailto:hongji.zhao@ericsson.com>

               Xufeng Liu
               <mailto:xufeng.liu.ietf@gmail.com>

               Yisong Liu
               <mailto:liuyisong@huawei.com>

               Mani Panchanathan
               <mailto:mapancha@cisco.com>

               Mahesh Sivakumar
               <mailto:sivakumar.mahesh@gmail.com>

    ";

  description
    "The module defines a collection of YANG definitions common for
     all Internet Group Management Protocol (IGMP) and Multicast
     Listener Discovery (MLD) Proxy devices.

     Copyright (c) 2019 IETF Trust and the persons identified as
     authors of the code.  All rights reserved.

     Redistribution and use in source and binary forms, with or
     without modification, is permitted pursuant to, and subject to
     the license terms contained in, the Simplified BSD License set
     forth in Section 4.c of the IETF Trust's Legal Provisions
     Relating to IETF Documents
     (http://trustee.ietf.org/license-info).

     This version of this YANG module is part of RFC XXXX; see the
     RFC itself for full legal notices.";

  revision 2019-07-03 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for IGMP and MLD Proxy";
  }


  /*
   * Features
   */
```

```
  feature feature-igmp-proxy {
    description
      "Support IGMP Proxy protocol.";
    reference
      "RFC 4605";
  }

  feature feature-mld-proxy {
    description
      "Support MLD Proxy protocol.";
    reference
      "RFC 4605";
  }

  /*
   * Identities
   */

  identity igmp-proxy {
    base rt:control-plane-protocol;
    description
      "IGMP Proxy protocol";
  }

  identity mld-proxy {
    base rt:control-plane-protocol;
    description
      "MLD Proxy protocol";
  }


  /*
   * Typedefs
   */


  /*
   * Groupings
   */

  grouping per-interface-config-attributes {

    description "Config attributes under interface view";

    leaf enable {
      type boolean;
      default false;
      description
        "Set the value to true to enable IGMP/MLD proxy";
    }
```

```
  } // per-interface-config-attributes

  grouping state-group-attributes {
    description
      "State group attributes";

    leaf up-time {
      type uint32;
      units seconds;
      description
        "The elapsed time for (S,G) or (*,G).";
    }

    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.";
    }
  } // state-group-attributes


 /* augments */

  augment "/rt:routing/rt:control-plane-protocols"+
        "/rt:control-plane-protocol" {

    description
      "IGMP Proxy augmentation to routing control plane protocol
       configuration and state.";

    container igmp-proxy {
     when 'derived-from-or-self(../rt:type, "imp:igmp-proxy")' {
        description
          "This container is only valid for IGMP Proxy protocol.";
      }
```

```
      if-feature feature-igmp-proxy;
      description "IGMP proxy";
      container interfaces {
        description
          "Containing a list of upstream interfaces.";

        list interface {
          key "interface-name";
          description
            "List of upstream interfaces.";


          leaf interface-name {
            type if:interface-ref;
            must "not( current() = /rt:routing"+
           "/rt:control-plane-protocols/pim-base:pim"+
           "/pim-base:interfaces/pim-base:interface"+
           "/pim-base:name )" {

           description
                "The upstream interface for IGMP proxy
                 should not be configured PIM.";
            }
            description "The upstream interface name.";
          }

          leaf version {
            type uint8 {
              range "1..3";
            }
            default 2;
            description "IGMP version.";
          }

          uses per-interface-config-attributes;

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

            leaf group-address {
              type inet:ipv4-address;
              description
                "Multicast group address.";
            }

            uses state-group-attributes;
```

```
            list source {
              key "source-address";
              description
                "List of multicast source information
                 of the multicast group.";
              leaf source-address {
                type inet:ipv4-address;
                description
                  "Multicast source address";
              }

              uses state-group-attributes;

              list downstream-interface {
                key "interface-name";
                description "The downstream interfaces list.";
                leaf interface-name {
                  type if:interface-ref;
                  description
                    "Downstream interfaces for each upstream-interface";
                }
            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.";
                }
              }
            } // list source
          } // list group
        } // interface
      } // interfaces
    }
  }
```

```
  augment "/rt:routing/rt:control-plane-protocols"+
        "/rt:control-plane-protocol" {

    description
      "MLD Proxy augmentation to routing control plane protocol
       configuration and state.";

    container mld-proxy {
      when 'derived-from-or-self(../rt:type, "imp:mld-proxy")' {
        description
          "This container is only valid for MLD Proxy protocol.";
      }
      if-feature feature-mld-proxy;
      description "MLD proxy";
      container interfaces {
        description
          "Containing a list of upstream interfaces.";

        list interface {
          key "interface-name";
          description
            "List of upstream interfaces.";

          leaf interface-name {
            type if:interface-ref;
            must "not( current() = /rt:routing"+
           "/rt:control-plane-protocols/pim-base:pim"+
           "/pim-base:interfaces/pim-base:interface"+
           "/pim-base:name )" {

           description
                "The upstream interface for MLD proxy
                 should not be configured PIM.";
            }
           description "The upstream interface name.";
          }

          leaf version {
            type uint8 {
              range "1..2";
            }
            default 2;
            description "MLD version.";
          }

          uses per-interface-config-attributes;

          list group {
            key "group-address";
            config false;
            description
```

```
              "Multicast group membership information
              that joined on the interface.";

            leaf group-address {
              type inet:ipv6-address;
              description
                "Multicast group address.";
            }

            uses state-group-attributes;

            list source {
              key "source-address";
              description
                "List of multicast source information
                 of the multicast group.";
              leaf source-address {
                type inet:ipv6-address;
                description
                  "Multicast source address";
              }

              uses state-group-attributes;

              list downstream-interface {
                key "interface-name";
                description "The downstream interfaces list.";
                leaf interface-name {
                  type if:interface-ref;
                  description
                    "Downstream interfaces for each upstream-interface";
                }
            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.";
                }
              }
            } // list source
          } // list group
        } // interface
      } // interfaces
    }
  }

 /*  RPCs  */

}
<CODE ENDS>
```

4. Security Considerations

The YANG module specified in this document defines a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC5246].

The NETCONF access control model [RFC6536] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in this YANG module that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) to these data nodes without proper protection can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/vulnerability:

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol

Unauthorized access to any data node of these subtrees can adversely affect the IGMP/MLD proxy subsystem of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations, and other problems.

Some of the readable data nodes in this YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol

Unauthorized access to any data node of these subtrees can disclose the operational state information of IGMP/MLD proxy on this device.

5. 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-proxy

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-proxy

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

prefix:       imp

reference:    RFC XXXX

--------------------------------------------------------------------
```

6. Normative References

[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.

[RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, August 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.

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

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

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, July 2013.

[RFC8342] M. Bjorklund and J. Schoenwaelder, "Network Management Datastore Architecture (NMDA)", RFC 8342, March 2018.

[RFC8343] M. Bjorklund, "A YANG Data Model for Interface Management", RFC 8343, March 2018.

[draft-ietf-pim-igmp-mld-yang-06] X. Liu, F. Guo, M. Sivakumar, P. McAllister, A. Peter, "A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)", draft-ietf-pim-igmp-mld-yang-06, Oct 20, 2017.

[draft-dsdt-nmda-guidelines-01] M. Bjorklund, J. Schoenwaelder, P. Shafer, K. Watsen, R. Wilton, "Guidelines for YANG Module Authors (NMDA)", draft-dsdt-nmda-guidelines-01, May 2017 [draft-ietf-netmod-revised-datastores-03] M. Bjorklund, J. Schoenwaelder, P. Shafer, K. Watsen, R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-03, July 3, 2017

Authors' Addresses

Hongji Zhao
Ericsson (China) Communications Company Ltd.
Ericsson Tower, No. 5 Lize East Street,
Chaoyang District Beijing 100102, P.R. China
Email: hongji.zhao@ericsson.com

Xufeng Liu
Volta Networks
USA
EMail: Xufeng.liu.ietf@gmail.com

Yisong Liu
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China
Email: liuyisong@huawei.com

Mani Panchanathan
Cisco
India
Email: mapancha@cisco.com

Mahesh Sivakumar
Juniper Networks
1133 Innovation Way
Sunnyvale, California
USA
EMail: sivakumar.mahesh@gmail.com