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’s stability with respect to packet loss and convergence time when the PIM Designated Router (DR) role changes.
Multicast technology, with PIM-SM ([RFC7761]), is used widely in Modern services. 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). Also, the PIM DR is responsible for tracking local multicast listeners and forwarding data to these listeners.
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. The failure detection and election procedures may 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 be elected as DR. RouterA will stop forwarding multicast packets, and the flows will not recover until RouterB assumes them.

In either of the situations listed, many multicast packets may be lost, and the quality of the 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 the identity of these nodes 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

Modern services: The real time multicast services, such like IPTV, Net-meeting, etc.
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.

Sticky: The DR doesn’t change unnecessarily when routers, even with higher priority, go down or come up.

3. Protocol Specification

The router follows the following procedures, these steps are to be used when a router starts, or the interface is enabled:

(a). When a router first starts or its interface is enabled, it includes the DR and BDR Address options with the OptionValue set to 0x0 in its Hello messages (Section 4). At this point the router considers itself a DROther, and starts a timer set to Default_Hello_Holdtime [RFC7761].

(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 address.

(c). After the timer expires, the router first executes the algorithm defined in section 3.1. After that, the router acts as one of the roles in the LAN: DR, BDR, or DROther.

If the router is elected the BDR, it takes on all the functions of a DR as specified in [RFC7761], but it SHOULD NOT actively forward multicast flows or send a register message to avoid duplication.

If the DR becomes unreachable on the LAN, the BDR MUST take over all the DR functions, including multicast flow forwarding and sending the Register messages. Mechanisms outside the scope of this specification, such as [RFC9186] 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, B joins; C is still the DR, and B is the BDR. Later A joins, then A becomes the BDR, and B is simply DROther.

3.1. Election Algorithm

The DR and BDR election refers the DR election algorithm defined in section 9.4 in [RFC2328], and updates the election function defined in section 4.3.2 in [RFC7761].

* The DR is elected among the DR candidates directly. If there is no DR candidates, i.e., all the routers advertise the DR Address options with zero OptionValue, the elected BDR will be the DR. And then the BDR is elected again from the other routers in the LAN.

* 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 for the elected DR, is elected as the BDR. That is the BDR may be the router which has the highest priority in the LAN.

* The advertisement is through PIM Hello message.

Except for the information recorded in section 4.3.2 in [RFC7761], the DR/BDR OptionValue from the neighbor is also recorded:

* neighbor.dr: The DR Address OptionValue that presents in the Hello message from the PIM neighbor.

* neighbor.bdr: The BDR Address OptionValue that presents in the Hello message from the PIM neighbor.

The pseudocode is shown below: A BDR election function is added, and the DR function is updated. The validneighbor function means that a valid Hello message has been received from this neighbor.
BDR(I) {
    bdr = NULL
    for each neighbor on interface I {
        if (neighbor.bdr != NULL) {
            if (validneighbor(neighbor.bdr) == TRUE) {
                if bdr == NULL
                    bdr = neighbor.bdr
                else if dr_is_better(neighbor.bdr, bdr, I) == TRUE
                        bdr = neighbor.bdr
            }
        }
    }
    return bdr
}

DR(I) {
    dr = NULL
    for each neighbor on interface I {
        if (neighbor.dr != NULL) {
            if (validneighbor(neighbor.dr) == TRUE) {
                if (dr == NULL)
                    dr = neighbor.dr
                else if dr_is_better(neighbor.dr, dr, I) == TRUE
                        dr = neighbor.dr
            }
        }
    }
    if (dr == NULL) {
        dr = bdr
    }
    if (dr == NULL) {
        dr = me
    }
    return dr
}

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

* 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 OptionValue of the DR Address option set to 0x0. The BDR Address option SHOULD also be sent, the OptionValue MUST be set to 0x0. Then the interface starts a timer which value is set to Default_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).

After the election, if there is one existed DR in the LAN, the DR remains unchanged. If there is no existed DR in the LAN, a new DR is elected, the routers in the LAN MUST send the Hello message with the OptionValue of DR Address option set to the elected DR. If there are more than one routers with non-zero DR priority in the LAN, a BDR is also elected. Then the routers in the LAN MUST send the Hello message with the OptionValue of BDR Address option set to the elected BDR. Any DROther router MUST NOT use its IP addresses in the DR/BDR Address option.
For example, there is a stable LAN that includes RouterA and RouterB. RouterA is the DR that has the highest priority. RouteC is a newcomer. RouterC sends a Hello message with the OptionValue of DR/BDR Address option set to zero. RouterA and RouterB sends the Hello message with the DR OptionValue set to RouterA, the BDR OptionValue set to RouterB.

In case RouterC has a higher priority than RouterB, RouterC elects itself as the BDR after it runs the election algorithm, then RouterC sends Hello messages with the DR OptionValue set to the IP address of current DR (RouterA), and the BDR OptionValue set to RouterC.

In case RouterB has a higher priority than RouterC, RouterC finds that it can not be the BDR after it runs the election algorithm, it sets the status to DROther. Then RouterC sends Hello messages with the DR OptionValue set to RouterA and the BDR OptionValue set to RouterB.

3.3. Receiving Hello Messages

When a Hello message is received, the OptionValue of DR/BDR is checked. If the OptionValue of DR is not zero and it isn’t the same with local stored values, or the OptionValue of DR is zero but the advertising router is the stored DR, the interface timer of election MAY be set/reset.

Before the election algorithm runs, the validity check MUST be done. The DR/BDR OptionValue in the Hello message MUST match with a known neighbor, otherwise the DR/BDR OptionValue can not become the DR/BDR candidates.

If there is one or more candidates which are different from the stored DR/BDR value after the validity check, the election MUST be taken. The new DR/BDR will be elected according to the rules defined in section 3.1.
3.4. Working with the DRLB function

A network can use the enhancement described in this document with the DR Load Balancing (DRLB) mechanism [RFC8775]. The DR MUST send the DRLB-List Hello Option defined in [RFC8775]. If the DR becomes unreachable, the BDR will take over all the multicast flows on the link, which may result in duplicated traffic as it may not have been a Group DR (GDR). The new DR MUST then follow the procedures in [RFC8775].

In case the DR, or the BDR which becomes DR after the DR failure, doesn’t support the mechanism defined in [RFC8775], the DRLB-List Hello Option cannot be advertised, then the DRLB mechanism takes no effect.

4. PIM Hello message format

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

![Hello Option Format](image)

4.1. DR Address Option format

![DR Address Option](image)

* OptionType: The value is 37.

**DR Address**: 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

```
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| Type = 38 | Length = <Variable> |
| +-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
| BDR Address (Encoded-Unicast format) |
| +-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+-----------------+
```

**Figure 5: BDR Address Option**

**OptionType**: The value is 38.

**OptionLength**: 4 bytes if using IPv4 and 16 bytes if using IPv6.

**BDR Address**: 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 correspond to an address used to send PIM Hello messages by one of the PIM neighbors on the interface. If that is not the case then the OptionValue of DR/BDR MUST be ignored as described in section 3.3.

An option with unexpected values MUST be ignored. For example, a DR Address option with an IPv4 address received while the interface only supports IPv6 is ignored.

### 5. Backwards Compatibility

Any router using the DR and BDR Address Options MUST set the corresponding OptionValues. If at least one router on a LAN doesn’t send a Hello message, including the DR Address Option, then the specification in this document MUST NOT be used. For example, the routers in a LAN all support the options defined in this document, the DR/BDR is elected. A new router which doesn’t support the options joins, when the hello message without DR Address Option is received, all the router MUST switch the election function back.
immediately. This action results in all routers using the DR election function defined in [RFC7761] or [I-D.ietf-pim-bdr]. Both this draft and the draft [I-D.ietf-pim-bdr], introduce a backup DR. The later draft does this without introducing new options but does not consider the sticky behavior. In case there is router which doesn’t support the DR/BDR Address Option defined in this document, the routers SHOULD take the function defined in [I-D.ietf-pim-bdr] if all the routers support it, otherwise the router SHOULD used the function defined in [RFC7761].

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.

6. Security Considerations

[RFC7761] describes the security concerns related to PIM-SM. A rogue router can become the DR/BDR by appropriately crafting the Address options to include a more desirable IP address or priority. Because the election algorithm makes the DR role be non-preemptive, an attacker can then take control for long periods of time. The effect of these actions can result in multicast flows not being forwarded (already considered in [RFC7761]).

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 untrusted 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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>37</td>
<td>DR Address Option</td>
<td>This Document</td>
</tr>
<tr>
<td>38</td>
<td>BDR Address Option</td>
<td>This Document</td>
</tr>
</tbody>
</table>

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


9.2. Informative References


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
A Yang Data Model for IGMP and MLD Snooping
draft-ietf-pim-igmp-mld-snooping-yang-20.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

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

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

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

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

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

This Internet-Draft will expire on April 07, 2022.
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].

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.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>YANG module</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>inet</td>
<td>ietf-inet-types</td>
<td>[RFC6991]</td>
</tr>
<tr>
<td>yang</td>
<td>ietf-yang-types</td>
<td>[RFC6991]</td>
</tr>
<tr>
<td>if</td>
<td>ietf-interfaces</td>
<td>[RFC8343]</td>
</tr>
<tr>
<td>rt</td>
<td>ietf-routing</td>
<td>[RFC8349]</td>
</tr>
<tr>
<td>rt-types</td>
<td>ietf-routing-types</td>
<td>[RFC8294]</td>
</tr>
<tr>
<td>dot1q</td>
<td>ieee802-dot1q-bridge</td>
<td>[dot1Qcp]</td>
</tr>
</tbody>
</table>

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]. It also includes actions for clearing IGMP and MLD Snooping group tables.

The YANG module doesn’t cover L2VPN, which will be specified in a separated document.

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

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, and it corresponds to one BRIDGE instance.

Currently the value of l2-service-type in igmp-snooping-instance could only be set bridge. After it is set, igmp-snooping-instance could be used in the BRIDGE service.

The values of bridge-mrouter-interface is filled by the snooping device dynamically. It is different from static-bridge-mrouter-interface which is configured.

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

```yaml
augment /rt:routing/rt:control-plane-protocols
   /rt:control-plane-protocol:
   +--rw igmp-snooping-instance {igmp-snooping}?
      +--rw 12-service-type? 12-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 fast-leave? empty {fast-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 igmp-version? uint8
      +--rw querier-source? inet:ipv4-address
      +--rw static-l2-multicast-group* [group source-addr]
      |       {static-l2-multicast-group}?
      |       rt-types:ipv4-multicast-group-address
      +--rw source-addr
      |       rt-types:ipv4-multicast-source-address
      +--rw bridge-outgoing-interface* if:interface-ref
      |       entries-count? yang:gauge32
      +--ro bridge-mrouter-interface* if:interface-ref
      +--ro group* [address]
      |       +--ro address
```
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] 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, and it corresponds to one BRIDGE instance.

Currently the value of l2-service-type in mld-snooping-instance could only be set bridge. After it is set, mld-snooping-instance could be used in the BRIDGE service.

The value of bridge-mrouter-interface is filled by the snooping device dynamically. It is different from static-bridge-mrouter-interface which is configured.

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

```yaml
augment /rt:routing/rt:control-plane-protocols
    /rt:control-plane-protocol:
        +--rw mld-snooping-instance {mld-snooping}?
            +--rw l2-service-type? 12-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 fast-leave? empty {fast-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 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
                    +--ro entries-count? yang:gauge32
                    +--ro bridge-mrouter-interface* if:interface-ref
                    +--ro group* [address]
```
3.3. Using IGMP and MLD Snooping Instances

The igmp-snooping-instance could be used in the service of BRIDGE [dot1Qcp] 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.
The mld-snooping-instance could be used in concurrence with igmp-snooping-instance to configure the MLD Snooping.

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

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],[dot1Qcp].

<CODE BEGINS> file ietf-igmp-mld-snooping@2021-10-08.yang
module ietf-igmp-mld-snooping {  
  yang-version 1.1;
  prefix ims;
  import ietf-inet-types {  
    prefix "inet";  
  }
  
Zhao & Liu, etc Expires April 07, 2022 [Page 11]
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 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>
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) 2021 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 2021-10-08 {
    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 fast-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 static L2 multicast 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 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 "#/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
  "When it is true, this switch will send out periodic IGMP General Query Message or MLD General Query Message."
}

leaf fast-leave {
  if-feature fast-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. If it doesn’t exist,
         the IGMP or MLD packet will be ignored."
        reference "RFC 3376. Sec. 9.1, 9.2, 9.3.";
}

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";
}
} // 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 dynamically 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.";
  }
} // 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 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

Zhao & Liu, etc Expires April 07, 2022 [Page 21]
"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
      "It indicates BRIDGE or other services.";
  }

  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
    "The source address of IGMP General Query message,
    which is sent out by this switch."
}

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"
  }
}

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 "address";
  description
    "List of multicast membership hosts
    of the specific multicast source-group.";
  leaf address {
    type inet:ipv4-address;
    description
      "Multicast membership host address.";
  }
  leaf 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
   "It indicates BRIDGE or other services."
}

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
      "The source address of MLD General Query message,
       which is sent out by this switch."
}

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";
   }
// 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 "address";
description
    "List of multicast membership hosts
    of the specific multicast source-group.";

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

  leaf 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 {

}
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";
    }
} // augment
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:
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:

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:
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]:

```
---
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
prefix:  ims
reference: RFC XXXX
---
```
7. References

7.1. Normative References


Internet-Draft     IGMP & MLD Snooping Yang Module     October 08, 2021

[ 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.

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.

RFC 7950, August 2016.


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

8340, March 2018.


RFC 8343, March 2018.

Management (NMDA Version)", RFC 8349, March 2018.

Version 1.3", RFC 8446, August 2018.

7.2. Informative References

[ 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.

Zhao & Liu, etc           Expires April 07, 2022              [Page 35]
Appendix A. Data Tree Example

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

```
+-----------+
|  Source   |
+-----+-----+

.JSONObjectExample

|eth1/1
+-------+
|  R1    |
+-------+

|eth1/2 | eth1/3
+-------+

|eth2/1 | eth3/1
+-------+-------+
|  R2    |  R3    |
+-------+-------+

|eth2/2 | eth3/2
+-------+

```

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

```json
{
  "ietf-interfaces:interfaces":{
    "interface":[
      {
        "name":"eth1/1",
        "type":"iana-if-type:ethernetCsmacd"
      }
    ]
  }
}
```
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"
      }
    }]
  }
}
```
The following action is to clear all the entries whose group address is 225.1.1.1 for igmp-snooping-instance bis1.

Host: example.com
Content-Type: application/yang-data+json
{"ietf-igmp-mld-snooping:input": {
    "group": "225.1.1.1",
    "source": "**"
}}

Authors' Addresses

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

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

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

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

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

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

This Internet-Draft will expire on December 14, 2019.

Copyright Notice

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

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

Abstract

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

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...................................................3  
   1.1. Terminology...............................................3  
   1.2. Tree Diagrams.............................................4  
   1.3. Prefixes in Data Node Names...............................4  
2. Design of Data model...........................................4  
   2.1. Scope of Model............................................4  
   2.1.1. Parameters Not Covered at Global Level...............5  
   2.1.2. Parameters Not Covered at Interface Level............5  
   2.2. Optional Capabilities.....................................6  
   2.3. Position of Address Family in Hierarchy................6  
3. Module Structure...............................................7  
   3.1. IGMP Configuration and Operational State.................7  
   3.2. MLD Configuration and Operational State................10  
   3.3. IGMP and MLD Actions.....................................13  
4. IGMP and MLD YANG Module......................................13  
5. Security Considerations.......................................43  
6. IANA Considerations...........................................45  
7. Acknowledgments...............................................46  
8. Contributing Authors..........................................46  
9. References....................................................46  
   9.1. Normative References.....................................46  
   9.2. Informative References...................................48
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.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>YANG module</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>yang</td>
<td>ietf-yang-types</td>
<td>[RFC6991]</td>
</tr>
<tr>
<td>inet</td>
<td>ietf-inet-types</td>
<td>[RFC6991]</td>
</tr>
<tr>
<td>if</td>
<td>ietf-interfaces</td>
<td>[RFC8343]</td>
</tr>
<tr>
<td>ip</td>
<td>ietf-ip</td>
<td>[RFC8344]</td>
</tr>
<tr>
<td>rt</td>
<td>ietf-routing</td>
<td>[RFC8349]</td>
</tr>
<tr>
<td>rt-types</td>
<td>ietf-routing-types</td>
<td>[RFC8294]</td>
</tr>
<tr>
<td>acl</td>
<td>ietf-access-control-list</td>
<td>[RFC8519]</td>
</tr>
</tbody>
</table>

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
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:

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

- The structure is consistent with other YANG models such as RFC 8344, which uses separate branches for IPv4 and IPv6.
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 
      |       |       --> /acl:acls/acl/name {intf-source-policy}?
        |       ++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
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]
| {intf-source-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]
| {intf-source-address}            |
++-ro source-address              inet:ipv6-address
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], [RFC3579], [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";

Liu & Guo, etc Expires December, 2019 [Page 14]
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.";
}
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.";
}
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;
    }
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

//interface-common-config-attributes

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
    "If present, the interface accepts packets with matching source IP subnet only."
}
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."
      }
    }
  }

} // interface-config-attributes-mld
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
  }
} // interface-state-attributes
"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
} // 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.";
  }

}

Liu & Guo, etc Expires December, 2019 [Page 34]
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 " + "source specific parameters.";
      }
    }
  }
}
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.";
                }
            }
        }
    }
} // actions
leaf group-address {
  type union {
    type enumeration {
      enum '*' {
        description "Any group address.";
      }
    }
  }
  type rt-types:ipv4-multicast-group-address;
}

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 "/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.";
    }
  }
}

Liu & Guo, etc Expires December, 2019
must ".*/version = 2 or "+ "!(./.*/version) and "+ "(.*/.*/version = 2 or !(./.*/version))"

error-message
"The version of MLD must be 2 to support the "
+ "source specific parameters.";
"
}

refine source-policy {
must ".*/version = 2 or "+ "!(./.*/version) and "+ "(.*/.*/version = 2 or !(./.*/version))"

error-message
"The version of MLD must be 2 to support the "
+ "source specific parameters.";
"
}

refine explicit-tracking {
must ".*/version = 2 or "+ "!(./.*/version) and "+ "(.*/.*/version = 2 or !(./.*/version))"

error-message
"The version of MLD must be 2 to support the "
+ "explicit tracking function.";
"
}

refine lite-exclude-filter {
must ".*/version = 2 or "+ "!(./.*/version) and "+ "(.*/.*/version = 2 or !(./.*/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.";
  }
}
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/igmp-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]:

--------------------------------------------------------------------
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
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


9.2. Informative References


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
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.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 13, 2019.

Copyright Notice

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

- The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding provider’s multicast router.

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

- 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 assessed 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

- 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.
- 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.
- 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

- 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.

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

- 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).

- 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 policies 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

- 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.

- 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 to 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.

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

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Multicast Wholesale</th>
<th>Multicast Resiliency</th>
<th>Load Balancing</th>
<th>Network Merging</th>
<th>Network Migration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upstream Control Delivery</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Downstr. Control Delivery</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Active / Standby Upstream</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upstr i/f selection per group</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upstr i/f selection all group</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>ASM</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>SSM</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

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


9.2. Informative References


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
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

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

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

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

This Internet-Draft will expire on April 25, 2019.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents
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|         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.
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
| +-------------------------------------------------+
| | Type = TBD | Length |
| +-------------------------------------------------+
| | Group Address (Encoded-Group format) |
| +-------------------------------------------------+
| | Src Count | Src Holdtime |
| +-------------------------------------------------+
| | Src Address 1 (Encoded-Unicast format) |
| +-------------------------------------------------+
| | Metric Preference 1 |
| +-------------------------------------------------+
| | Src Address 2 (Encoded-Unicast format) |
| +-------------------------------------------------+
| | Metric Preference 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].
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

8.2. Informative References

[I-D.ietf-bier-mld]

[I-D.ietf-bier-pim-signaling]

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
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).

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt
Internet-Draft        IGMP/MLD Proxy Yang Module          July 03, 2019

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

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

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

Table of Contents

1. Introduction...................................................3
   1.1. Terminology...............................................3
   1.2. Tree Diagrams.............................................3
2. Design of Data Model...........................................3
   2.1. Overview..................................................4
   2.2. Augment /rt:routing/rt:control-plane-protocols/rt:control-
        plane-protocol.................................................4
3. IGMP/MLD Proxy YANG Module.....................................5
4. Security Considerations.......................................13
5. IANA Considerations...........................................14
6. Normative References..........................................15
Authors’ Addresses...............................................17

Zhao & Liu, etc        Expires January 02, 2020                [Page 2]
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:

- Brackets "[" and "]" enclose list keys.
- Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).
- Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- 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

Zhao & Liu, etc  Expires January 02, 2020
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;
  // 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";

Zhao & Liu, etc Expires January 02, 2020
Internet-Draft        IGMP/MLD Proxy Yang Module          July 03, 2019

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
 */

Zhao & Liu, etc Expires January 02, 2020 [Page 6]
feature feature-igmp-proxy {
  description "Support IGMP Proxy protocol.";
  reference "RFC 4605";
}

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

дать 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";
  }
}
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.";
  }
}

/* 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.";
    }
  }

  Zhao & Liu, etc Expires January 02, 2020 [Page 8]
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
Z
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
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]:

| 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
| prefix: imp
| reference: RFC XXXX
6. Normative References


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 Beijing 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