A YANG Data Model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)
draft-ietf-pim-igmp-mld-yang-07

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 22, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
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 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. This model will support the core IGMP and MLD protocols, as well as many other features.
mentioned in separate IGMP and MLD RFCs. Non-core features are defined as optional in the provided data model.

1.1. Terminology

The terminology for describing YANG data models is found in [RFC6020] [RFC7950].

The following abbreviations are used in this document and the 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>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>rt</td>
<td>ietf-routing</td>
<td>[RFC8349]</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] and MLDv1 [RFC2710], MLDv2 [RFC3810].

The configuration of IGMP and MLD features, and the operational state fields and RPC definitions are not all included in this document of the data model. This model can be extended, though the structure of what has been written may be taken as representative of the structure of the whole model.

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

2.2. Optional capabilities

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

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

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

On the other hand, operational state parameters are not so widely designated as features, as there are many cases where the defaulting of an operational state parameter would not cause any harm to the system, and it is much more likely that an implementation without native support for a piece of operational state would be able to derive a suitable value for a state variable that is not natively supported.
For the same reason, wide constant ranges (for example, timer maximum and minimum) will be used in the model. It is expected that vendors will augment the model with any specific restrictions that might be required. Vendors may also extend the features list with proprietary extensions.

2.3. Position of address family in hierarchy

The current document contains IGMP and MLD as separate schema branches in the structure. The reason for this is to make it easier for implementations which may optionally choose to support specific address families. And the names of objects may be different between the IPv4 (IGMP) and IPv6 (MLD) address families.

3. Module Structure

3.1. IGMP Configuration and Operational state

The IGMP YANG model conforms to the Network Management Datastore Architecture (NMDA) [RFC8342]. The operational state data is combined with the associated configuration data in the same hierarchy [I-D.ietf-netmod-rfc6087bis]. The IGMP module defines in a three-level hierarchy structure as listed below:

Global level: IGMP configuration and operational state attributes for the entire routing system.

Interface-global: Only including configuration data nodes that IGMP configuration attributes are applicable to all the interfaces whose interface-level corresponding attributes are not existing, with same attributes’ value for these interfaces.

Interface-level: IGMP configuration and operational state attributes specific to the given interface.

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

This model augments the core routing data model "ietf-routing" specified in [RFC8349]. The IGMP model augments "/rt:routing/rt:control-plane-protocols" as opposed to augmenting "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol", as the latter would allow multiple protocol instances, while the IGMP protocol is designed to be enabled or disabled as a single protocol instance on a network instance or a logical network element.

augment /rt:routing/rt:control-plane-protocols:
    +--rw 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 max-groups-per-interface?     uint32 {intf-max-groups}?
    +++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 interface* [interface-name]
      +++rw interface-name         if:interface-ref
      +++rw enable?               boolean {intf-admin-enable}?
      +++rw group-policy?         String
      +++rw immediate-leave?      empty {intf-immediate-leave}?
      +++rw last-member-query-interval?   uint16
      +++rw max-groups?           uint32 {intf-max-groups}?
      +++rw max-group-sources?    uint32 {intf-max-group-sources}?
      +++rw query-interval?       uint16
      +++rw query-max-response-time?      uint16
      +++rw require-router-alert? boolean {intf-require-router-alert}?
      +++rw robustness-variable?  uint8
      +++rw source-policy?        string {intf-source-policy}?
      +++rw verify-source-subnet? empty {intf-verify-source-subnet}?
      +++rw explicit-tracking?    boolean {intf-explicit-tracking}?
      +++rw exclude-lite?         boolean {intf-exclude-lite}?
3.2. MLD Configuration and Operational State

The MLD YANG model uses the same structure as IGMP YANG model. The MLD module also defines in a three-level hierarchy structure as listed below:

```
augment /rt:routing/rt:control-plane-protocols:
  +++-rw 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
```
++rw interfaces
  ++rw last-member-query-interval?   uint16
  ++rw max-groups-per-interface?    uint32 {intf-max-groups}?
  ++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 interface* [interface-name]
  ++rw interface-name            if:interface-ref
  ++rw enable?                   boolean {intf-admin-enable}?
  ++rw group-policy?             string
  ++rw immediate-leave?          empty {intf-immediate-leave}
++rw last-member-query-interval?   uint16
++rw max-groups?                uint32 {intf-max-groups}?
++rw max-group-sources?         uint32 {intf-max-group-sources}?
++rw query-interval?            uint16
++rw query-max-response-time?   uint16
++rw require-router-alert?      boolean {intf-require-router-alert}?
++rw robustness-variable?       uint8
++rw source-policy?             string {intf-source-policy}?
++rw verify-source-subnet?      empty {intf-verify-source-subnet}?
++rw explicit-tracking?         boolean {intf-explicit-tracking}?
++rw exclude-lite?              boolean {intf-exclude-lite}?
++rw version?                   uint8
++rw join-group*                inet:ipv6-address {intf-join-group}?
++rw ssm-map* [source-addr group-policy] {intf-ssm-map}?
  ++rw source-addr               ssm-map-ipv6-addr-type
  ++rw group-policy              string
++rw static-group* [group source-addr] {intf-static-group}?
  ++rw group                    inet:ipv6-address
  ++rw source-addr              source-ipv6-addr-type
++ro oper-status?               enumeration
++ro querier?                   inet:ipv6-address
++ro joined-group*              inet:ipv6-address {intf-join-group}?
3.3. IGMP and MLD RPC

IGMP and MLD RPC clears the specified IGMP and MLD group membership.

rpcs:

```yml
---x clear-igmp-groups {rpc-clear-groups}?
  | +---w input
  |   | +---w interface? string
  |   | +---w group? inet:ipv4-address
  |   | +---w source? inet:ipv4-address

---x clear-mld-groups {rpc-clear-groups}?
  | +---w input
  |   | +---w interface? string
  |   | +---w group? inet:ipv6-address
  |   | +---w source? inet:ipv6-address
```

4. IGMP and MLD YANG Modules

```yml
<CODE BEGINS> file "ietf-igmp-mld@2018-06-22.yang"
module ietf-igmp-mld {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld";
}  
<CODE ENDS>
```
// replace with IANA namespace when assigned
prefix igmp-mld;

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

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

import ietf-routing {
    prefix "rt";
}

import ietf-interfaces {
    prefix "if";
}

import ietf-ip {
    prefix ip;
}

organization
    "IETF PIM Working Group";

contact
    "WG Web: <http://tools.ietf.org/wg/pim/>
    WG List: <mailto:pim@ietf.org>
    WG Chair: Stig Venaas
    <mailto:stig@venaas.com>
    WG Chair: Mike McBride
    <mailto:mmcbride7@gmail.com>
    Editor: Xufeng Liu
    <mailto:xufeng.liu.ietf@gmail.com>
    Editor: Feng Guo
    <mailto:guofeng@huawei.com>
    Editor: Mahesh Sivakumar
    <mailto:masivaku@cisco.com>
    Editor: Pete McAllister
    <mailto:pete.mcallister@metaswitch.com>
    Editor: Anish Peter
The module defines a collection of YANG definitions common for IGMP and MLD.

revision 2018-06-22 {
  description
  "Updated yang data model for parameter range and description.";
  reference
  "RFC XXXX: A YANG Data Model for IGMP and MLD"
}
revision 2017-10-20 {
  description
  "Updated yang data model for adding explicit-tracking and lightweight IGMPv3 and MLDv2 function.";
  reference
  "RFC XXXX: A YANG Data Model for IGMP and MLD"
}
revision 2017-09-19 {
  description
  "Updated yang data model for NMDA version and errata.";
  reference
  "RFC XXXX: A YANG Data Model for IGMP and MLD"
}

/* Features
 * Features */
feature global-admin-enable {
  description
  "Support global configuration to enable or disable protocol.";
}
feature global-interface-config {
  description
  "Support global configuration applied for all interfaces.";
}
feature global-max-entries {
  description
  "Support configuration of global max-entries.";
}
feature global-max-groups {
  description
  "Support configuration of global max-groups.";
}
feature intf-admin-enable {
  description
    "Support configuration of interface administrative enabling.";
}

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

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

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

/*
 * Typedefs
 */
typedef ssm-map-ipv4-addr-type {
  type union {
    type enumeration {
      enum 'policy' {
        description
          "Source address is specified in SSM map policy.";
     }
    }
    type inet:ipv4-address;
  }
  description
    "Multicast source IP address type for SSM map.";
} // source-ipv4-addr-type

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

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

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

/*
 * Identities
 */

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

leaf enable {
  if-feature global-admin-enable;
  type boolean;
  description
  "true to enable IGMP or MLD in the routing instance;
   false to disable IGMP or MLD in the routing instance.";
}
leaf max-entries {
  if-feature global-max-entries;
  type uint32;
  description
    "The maximum number of entries in IGMP or MLD."
}
leaf max-groups {
  if-feature global-max-groups;
  type uint32;
  description
    "The maximum number of groups that IGMP
    or MLD can join."
}

// global-config-attributes

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

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

  leaf groups-count {
    type uint32;
    config false;
    description
      "The number of groups that IGMP or MLD joins."
  }

  container statistics {
    config false;
    description "Global statistics."

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

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

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

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

} // statistics
} // global-state-attributes

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

} // global-statistics-error

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

grouping interfaces-config-attributes {

description
"Configuration attributes applied to the interfaces whose per interface attributes are not existing."

leaf last-member-query-interval {

type uint16 {

range "1..65535";
}
units seconds;
default 1;
description
"Last Member Query Interval, which may be tuned to modify the leave latency of the network."
reference "RFC3376. Sec. 8.8."
}

leaf max-groups-per-interface {

if-feature intf-max-groups;
type uint32;
description
"The maximum number of groups that IGMP or MLD can join."
}

leaf query-interval {

type uint16 {

range "1..31744";
}
units seconds;
default 125;
description
"The Query Interval is the interval between General Queries sent by the Querier. In RFC3376, 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:
<table>
<thead>
<tr>
<th>mant</th>
<th>exp</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3 4 5 6 7</td>
<td></td>
</tr>
</tbody>
</table>
QQI = (mant | 0x10) << (exp + 3)
The maximum value of QQI is 31744.";

Liu & Guo, etc Expires December, 2018 [Page 17]
leaf query-max-response-time {
    type uint16 {
        range "1..65535";
    }
    units seconds;
    default 10;
    description "Query maximum response time specifies the maximum time allowed before sending a responding report.";
    reference "RFC3376. Sec. 4.1.1, 8.3, 8.14.3";
}

leaf require-router-alert {
    if-feature intf-require-router-alert;
    type boolean;
    default false;
    description "Protocol packets should contain router alert IP option.";
}

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 "RFC3376. Sec. 4.1.6, 8.1, 8.14.1";
}

} // interfaces-config-attributes

grouping interfaces-config-attributes-igmp {
    description "interfaces configuration for IGMP.";
    uses interfaces-config-attributes;
    leaf version {
        type uint8 {
            range "1..3";
        }
        description "IGMP version.";
        reference "RFC1112, RFC2236, RFC3376";
    }
}

reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2";
grouping interfaces-config-attributes-mld {
  description "interfaces configuration for MLD.";

  uses interfaces-config-attributes;
  leaf version {
    type uint8 {
      range "1..2";
    }
    description "MLD version."
    reference "RFC2710, RFC3810.";
  }
}

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

  uses interface-config-attributes-igmp-mld;

  leaf version {
    type uint8 {
      range "1..3";
    }
    description "IGMP version."
    reference "RFC1112, RFC2236, RFC3376.";
  }

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

  list ssm-map {
    if-feature intf-ssm-map;
    key "source-addr group-policy";
    description "The policy for (*,G) mapping to (S,G).";
    leaf source-addr {
      type ssm-map-ipv4-addr-type;
      description "Multicast source IP address.";
    }

    leaf group-policy {
      type string;
      description "Name of the access policy used to filter IGMP
      membership. A device can restrict the length
      and value of this name, possibly space and special
list static-group {
  if-feature intf-static-group;
  key "group-addr source-addr";
  description
    "A static multicast route, (*,G) or (S,G).";

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

  leaf source-addr {
    type source-ipv4-addr-type;
    description
      "Multicast source IP address.";
  }
}
} // interface-config-attributes-igmp

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

  leaf enable {
    if-feature intf-admin-enable;
    type boolean;
    default false;
    description
      "true to enable IGMP or MLD on the interface;
       false to disable IGMP or MLD on the interface.";
  }

  leaf group-policy {
    type string;
    description
      "Name of the access policy used to filter IGMP or MLD membership. A device can restrict the length
      and value of this name, possibly space and special characters are not allowed.";
  }

  leaf immediate-leave {
    if-feature intf-immediate-leave;
    type empty;
    description
      "If present, IGMP or MLD perform an immediate leave upon
       receiving an IGMP or MLD? leave message.";
  }
}
If the router is IGMP-enabled or MLD-enabled, it sends an IGMP or MLD last member query with a last member query response time. However, the router does not wait for the response time before it prunes off the group.

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

leaf max-groups {
  if-feature intf-max-groups;
  type uint32;
  description "The maximum number of groups that IGMP ro MLD can join.";
}

leaf max-group-sources {
  if-feature intf-max-group-sources;
  type uint32;
  description "The maximum number of (source, group) entries.";
}

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

leaf query-max-response-time {
  type uint16 {
    range "1..65535";
  }
  units seconds;
  default 10;
  description
    "Query maximum response time specifies the maximum time
    allowed before sending a responding report.";
  reference "RFC3376. Sec. 4.1.1, 8.3, 8.14.3.";
}

leaf require-router-alert {
  if-feature intf-require-router-alert;
  type boolean;
  description
    "Protocol packets should contain router alert IP option.";
}

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 "RFC3376. Sec. 4.1.6, 8.1, 8.14.1.";
}

leaf source-policy {
  if-feature intf-source-policy;
  type string;
  description
    "Name of the access policy used to filter sources.
    A device can restrict the length
    and value of this name, possibly space and special
    characters are not allowed.";
}

leaf verify-source-subnet {
  if-feature intf-verify-source-subnet;
  type empty;
  description
    "If present, the interface accepts packets with matching
    source IP subnet only.";
}
leaf explicit-tracking {
  if-feature intf-explicit-tracking;
  type boolean;
  description
    "IGMP/MLD-based explicit membership tracking function
    for multicast routers and IGMP/MLD proxy devices
    supporting IGMPv3/MLDv2. The explicit membership tracking
    function contributes to saving network resources and
    shortening leave latency."
}

leaf exclude-lite {
  if-feature intf-exclude-lite;
  type boolean;
  description
    "lightweight IGMPv3 and MLDv2 protocols, which simplify the
    standard versions of IGMPv3 and MLDv2.";
  reference "RFC5790";
}

} // interface-config-attributes-igmp-mld

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

  leaf version {
    type uint8 {
      range "1..2";
    }
    description "MLD version.";
    reference "RFC2710, RFC3810.";
  }

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

  list ssm-map {
    if-feature intf-ssm-map;
    key "source-addr group-policy";
    description "The policy for (*,G) mapping to (S,G).";
    leaf source-addr {
      type ssm-map-ipv6-addr-type;
      description
        "Multicast source IPv6 address.";
    }
  }

  } // interface-config-attributes-mld
leaf group-policy {
  type string;
  description
      "Name of the access policy used to filter MLD
      membership. 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 source-addr";
  description
      "A static multicast route, (*,G) or (S,G).";
  leaf group {
    type inet:ipv6-address;
    description
      "Multicast group IPv6 address."
  }
  leaf source-addr {
    type source-ipv6-addr-type;
    description
      "Multicast source IPv6 address."
  }
}

// interface-config-attributes-mld

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

  leaf querier {
    type inet:ipv4-address;
    config false;
    description "The querier address in the subnet";
  }
  leaf-list joined-group {
    if-feature intf-join-group;
    type inet:ipv4-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 inet:ipv4-address;
        description
            "Multicast group address."
    }
    uses interface-state-group-attributes-igmp-mld;

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

list source {
    key "source-address";
    description
        "List of multicast source information
        of the multicast group.";

    leaf source-address {
        type inet:ipv4-address;
        description
            "Multicast source address"
    }
    uses interface-state-source-attributes-igmp-mld;

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

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

    leaf host-address {
        type inet:ipv4-address;
        description
            "Multicast host address of
            the specific multicast group.";
    }
}
"Multicast membership host address."
}
leaf host-filter-mode {
  type enumeration {
    enum "include" {
      description
        "In include mode";
    }
    enum "exclude" {
      description
        "In exclude mode.";
    }
  }
  description
    "Filter mode for a multicast membership
    host may be either include or exclude.";
}
} // list host
} // list source
} // list group
} // interface-state-attributes-igmp

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

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

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

  uses interface-state-attributes-igmp-mld;
leaf querier {
  type inet:ipv6-address;
  config false;
  description
    "The querier address in the subnet.";
}
leaf-list joined-group {
  if-feature intf-join-group;
  type inet:ipv6-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 inet:ipv6-address;
    description
      "Multicast group address.";
  }
  uses interface-state-group-attributes-igmp-mld;
  leaf last-reporter {
    type inet:ipv6-address;
    description
      "The last host address which has sent the
      report to join the multicast group.";
  }
}
list source {
  key "source-address";
  description
    "List of multicast source information
    of the multicast group.";
  leaf source-address {
    type inet:ipv6-address;
    description
      "Multicast source address";
  }
  uses interface-state-source-attributes-igmp-mld;
  leaf last-reporter {
    type inet:ipv6-address;
  }
description
"The last host address which has sent the
report to join the multicast source and group.";
}
list host {
  if-feature intf-explicit-tracking;
  key "host-address";
  description
  "List of multicast membership hosts
  of the specific multicast source-group.";
  leaf host-address {
    type inet:ipv6-address;
    description
    "Multicast membership host address.";
  }
  leaf host-filter-mode {
    type enumeration {
      enum "include" {
        description
        "In include mode";
      }
      enum "exclude" {
        description
        "In exclude mode.";
      }
    }
    description
    "Filter mode for a multicast membership
    host may be either include or exclude.";
  }
} // list host
} // list source
} // interface-state-attributes-mld

grouping interface-state-group-attributes-igmp-mld {
  description
  "Per interface state attributes for both IGMP and MLD
groups.";
  leaf expire {
    type uint32;
    units seconds;
    description
    "The time left before multicast group 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.";
}

description
  "Filter mode for a multicast group, may be either include or exclude.";
leaf up-time {
  type uint32;
  units seconds;
  description
  "The elapsed time since the device created multicast group record.";
}
} // interface-state-group-attributes-igmp-mld

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

leaf expire {
  type uint32;
  units seconds;
  description
  "The time left before multicast source-group state expires.";
}
leaf up-time {
  type uint32;
  units seconds;
  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-igmp-mld
/*
 * Configuration and Operational state data nodes (NMDA version)
*/
augment "/rt:routing/rt:control-plane-protocols"
{
  description
  "IGMP augmentation to routing control plane protocol
  configuration and state."
;
  container igmp {
    description
    "IGMP 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 global-interface-config;
      }
;
      list interface {
        key "interface-name";
        description
        "List of IGMP interfaces."
;
        leaf interface-name {
          type if:interface-ref;
          must "/if:interfaces/if:interface[if:name = current()]/" + "ip:ipv4" {
            description
            "The interface must have IPv4 enabled."
;
            description
            "Reference to an entry in the global interface list."
;
            uses interface-config-attributes-igmp {
              if-feature per-interface-config;
            }
        }
    }
}
uses interface-state-attributes-igmp;
} // interface
} // interfaces
} // igmp
}//augment

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

container mld {
description
"MLD 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 global-interface-config;
}

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

    input {
        leaf interface {
            type string;
            description
               "Name of the IGMP interface.
               If it is not specified, groups from all interfaces are
               cleared.";
        }
        leaf group {
            type inet:ipv4-address;
            description
               "Multicast group IPv4 address.
               If it is not specified, all IGMP group entries are
               cleared.";
        }
        leaf source {
            type inet:ipv4-address;
            description
               "Multicast source IPv4 address.
               If it is not specified, all IGMP source-group entries are
               cleared.";
        }
    }
} // rpc clear-igmp-groups

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

    input {
        leaf interface {
            type string;
            description
               "Name of the MLD interface.
               If it is not specified, groups from all interfaces are
               cleared.";
        }
    }
} // rpc clear-mld-groups
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 [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:

igmp:global
This subtree specifies the configuration for the IGMP attributes at the global level on a device. Modifying the configuration can cause IGMP membership deleted or reconstructed on all the interfaces of a device.

igmp:interfaces

This subtree specifies the configuration for the IGMP attributes at all of the interfaces level on a device. Modifying the configuration can cause IGMP membership deleted or reconstructed on all the interfaces of a device.

igmp:interfaces/interface

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

These subtrees are all under /rt:routing/rt:control-plane protocols/igmp:

mld:global

This subtree specifies the configuration for the MLD attributes at the global level on a device. Modifying the configuration can cause MLD membership deleted or reconstructed on all the interfaces of a device.

mld:interfaces

This subtree specifies the configuration for the MLD attributes at all of the interfaces level on a device. Modifying the configuration can cause MLD membership deleted or reconstructed on all the interfaces of a device.

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 deleted or reconstructed on a specific interface of a device.

These subtrees are all under /rt:routing/rt:control-plane protocols/mld:

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/igmp

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

clear-igmp-groups

clear-mld-groups

Unauthorized access to any of the above RPC 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 [RFC7950]:
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


Liu & Guo, etc  Expires December, 2018
9.2. Informative References


[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-20(work in progress), March 2018
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-06

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.

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 20, 2018.

Copyright Notice

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

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

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.

```
+--------+  v  --------+
| Listener|  v  +--------+
+--------+  v  +--------+
          /         /         /
    O------( Multicast Set 1 )
          /         /         /
    O------( Multicast Set 2 )
          /         /         /
    O------( Multicast Set 3 )
        +--------+
          v
+--------+
| Proxy  |
+--------+
+--------+
| IGMP/MLD |
+--------+
```

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

This document is focused on both fixed and mobile network scenarios. Applicability of IGMP/MLD proxies with multiple upstream interfaces in mobile environments has been previously identified as beneficial in scenarios as the ones described in [RFC6224] and [RFC7287].
In the case of fixed networks, multicast wholesale services in a competitive residential market require an efficient distribution of multicast traffic from different operators or content providers, i.e. the incumbent operator and a number of alternative providers, on the network infrastructure of the former. Existing proposals are based on the use of PIM routing from the metro/core network, and multicast traffic aggregation on the same tree. A different approach could be achieved with the use of an IGMP/MLD proxy with multiple upstream interfaces, each of them pointing to a distinct multicast router in the metro/core border which is part of separated multicast trees deep in the network. Figure 2 graphically describes this scenario.

![Diagram of IGMP/MLD proxy with multiple upstream interfaces](image)

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

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

Having multiple upstream interfaces creates a new decision space for delivering the proper multicast content to the subscriber. Basically it is now possible to implement channel-based or subscriber-based 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.

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 can provide service flexibility in a lightweight and simpler manner if compared with PIM-routing based alternatives.

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 when a source S does not deliver a certain group G.

4.1.2. Multicast resiliency

In current PIM-based solutions, the resiliency of the multicast distribution relays on the routing capabilities provided by protocols like PIM 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.

4.1.2.1. Requirements

The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding active upstream interface.

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 typically forces the distribution of all the channels on the same path in the last segment of the network. Multiple upstream interfaces could naturally split the demand, alleviating the bandwidth requirements in the metro segment.
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.

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.

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

4.2. Mobile network scenarios

Mobile networks offer different alternatives for multicast distribution.

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

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

5. Summary of requirements

Following the analysis above, a number of different requirements can be identified by the IGMP/MLD proxy to support multiple upstream interfaces. The following table summarizes these requirements.
### Functionality 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</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Delivery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Downstream Control</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Delivery</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Active / Standby</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>Upstream</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upstr i/f selection</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>per group</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Upstr i/f selection</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
</tr>
<tr>
<td>all group</td>
<td></td>
<td></td>
<td></td>
<td></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>

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

### 6. Security Considerations

All the security considerations in [RFC4605] are directly applicable to this proposal. Apart from that, if proper mechanisms (i.e., implementation practices) are in place for channel-based or subscriber-based upstream interface selection, Denial of Service attacks can be prevented.
7. IANA Considerations

There are no IANA considerations.

8. Acknowledgements

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

9. References

9.1. Normative References


9.2. Informative References


Internet-Draft  Reqs for MLD proxy with multiple upstream  November 2017


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/
Abstract

Multi-Protocol BGP (MP-BGP) has support for distributing next-hop information for multiple address families using one AFI/Safi Network Layer Reachability Information (NLRI). [RFC5549] specifies the extensions necessary to allow advertising IPv4 NLRI or VPN-IPv4 NLRI with a Next Hop address that belongs to the IPv6 protocol. While the next-hop info is learnt via MP-BGP, certain procedures are needed to enable traffic forwarding. This document describes PIM extensions and the use-cases for multicast forwarding in various scenarios.
to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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

Table of Contents

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

1. Introduction

Figure 1: Example Topology

+-------------+                   +-------------+
|             |                   |             |
| Router1    1::1/64          1::2/64 Router2    |
| 10.1.1.1/32--+             +--------I1---------|             +-+PIM receiver
|         1.1.1.1/24         1.1.1.2/24         |
|             +                   +             |
|             |                   |             |
+-------------+                   +-------------+

While use of MP-BGP along with [RFC5549] enables one routing protocol session to exchange next-hop info for both IPv4 and IPv6 prefixes, forwarding plane needs additional procedures to enable forwarding in
data-plane. For example, when a IPv4 prefix is learnt over IPv6 next-hop, forwarding plane resolves the MAC-Address (L2-Adjacency) for IPv6 next-hop and uses it as destination-mac while doing inter-subnet forwarding. While it’s simple to find the required information for unicast forwarding, multicast forwarding in same scenario poses additional requirements.

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

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

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

2. Solution

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

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

3. Security Considerations

There are no new security considerations.

4. IANA Considerations

There are no IANA considerations.

5. References

5.1. Normative References


5.2. Informative References


Authors’ Addresses
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.

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 26, 2018.
1. Introduction

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

This data model follows the Guidelines for YANG Module Authors (NMDA)[draft-dsdt-nmda-guidelines-01]. 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) and Multicast Listener Discovery (MLD) Snooping Switches [RFC4541].

The goal of this document is to define a data model that provides a common user interface to IGMP and MLD Snooping. There is very information that is designated as "mandatory", providing freedom for vendors to adapt this data model to their respective product implementations.
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 protocol.

The YANG module includes IGMP and MLD Snooping instances definition, instance references in the scenario of BRIDGE, VPLS. The module also includes the RPC methods for clearing the specified IGMP and MLD Snooping.

This YANG model 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. IGMP and MLD Snooping Instances

The YANG module defines IGMP and MLD Snooping instance. The instance will be referenced in all kinds of scenarios to configure IGMP and MLD Snooping. The attribute who could be read and written shows configuration data. The read-only attribute shows state data. The key attribute is name.

```
module: ietf-igmp-mld-snooping

  +--rw igmp-snooping-instances
  |   +--rw igmp-snooping-instance* [name]
  |       +--rw name                        string
  |       +--rw id?                        uint32
  |       +--rw type?                     enumeration
  |       +--rw enable?                   boolean {admin-enable}?
  |       +--rw forwarding-mode?         enumeration
  |       +--rw explicit-tracking?       boolean {explicit-tracking}?
  |       +--rw exclude-lite?            boolean {exclude-lite}?

Zhao                    Expires April 26, 2018                 [Page 4]```
Internet-Draft     IGMP & MLD Snooping Yang Model     October 2017

|     +--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 version?    uint8
|     +--rw static-bridge-mrouter-interface*    if:interface-ref (static-l2-multicast-group)?
|     +--rw static-vpls-mrouter-interface*    12vpn-instance-pw-ref (static-l2-multicast-group)?
|     +--rw querier-source?    inet:ipv4-address
|     +--rw static-l2-multicast-group* [group source-addr] (static-l2-multicast-group)?
|     |     +--rw group    inet:ipv4-address
|     |     +--rw source-addr    source-ipv4-addr-type
|     |     +--rw bridge-outgoing-interface*    if:interface-ref
|     |     +--rw vpls-outgoing-ac*    12vpn-instance-ac-ref
|     |     +--rw vpls-outgoing-pw*    12vpn-instance-pw-ref
|     +--ro entries-count?    uint32
|     +--ro bridge-mrouter-interface*    if:interface-ref
|     +--ro vpls-mrouter-interface*    12vpn-instance-pw-ref
|     +--ro group* [address]
|     |     +--ro address    inet:ipv4-address
| | +-ro mac-address?  yang:phys-address
| | +-ro expire?  uint32
| | +-ro up-time?  uint32
| | +-ro last-reporter?  inet:ipv4-address
| | +-ro source* [address]
| |   | +-ro address  inet:ipv4-address
| |   | +-ro bridge-outgoing-interface*  if:interface-ref
| |   | +-ro vpls-outgoing-ac*  l2vpn-instance-ac-ref
| |   | +-ro vpls-outgoing-pw*  l2vpn-instance-pw-ref
| |   | +-ro up-time?  uint32
| |   | +-ro expire?  uint32
| |   | +-ro host-count?  uint32 {explicit-tracking}
| |   | +-ro last-reporter?  inet:ipv4-address
| | | +-ro host* [host-address] {explicit-tracking}?
| | |   | +-ro host-address  inet:ipv4-address
| | |   | +-ro host-filter-mode?  enumeration
| +-ro statistics
|   | +-ro received
| |   | +-ro query?  yang:counter64
| |   | +-ro membership-report-v1?  yang:counter64
| |   | +-ro membership-report-v2?  yang:counter64
| |   | +-ro membership-report-v3?  yang:counter64
| |   | +-ro leave?  yang:counter64
| |   | +-ro pim?  yang:counter64
|   | --ro sent
|   |   --ro query? yang:counter64
|   |   --ro membership-report-v1? yang:counter64
|   |   --ro membership-report-v2? yang:counter64
|   |   --ro membership-report-v3? yang:counter64
|   |   --ro leave? yang:counter64
|   |   --ro pim? yang:counter64

--rw mld-snooping-instances
| --rw mld-snooping-instance* [name]
|   | --rw name string
|   | --rw id? uint32
|   | --rw type? enumeration
|   | --rw enable? boolean {admin-enable}? 
|   | --rw forwarding-mode? enumeration
|   | --rw explicit-tracking? boolean {explicit-tracking }?
|   | --rw exclude-lite? boolean {exclude-lite}?
|   | --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 version? uint8
| ---rw static-bridge-mrouter-interface* if:interface-ref {static-l2-multicast-group}? |
| ---rw static-vpls-mrouter-interface* 12vpn-instance-pw-ref {static-l2-multicast-group}? |
| ---rw querier-source? inet:ipv6-address |
| ---rw static-l2-multicast-group* [group source-addr] {static-l2-multicast-group}? |
|   | ---rw group inet:ipv6-address |
|   | ---rw source-addr source-ipv6-addr-type |
|   | ---rw bridge-outgoing-interface* if:interface-ref |
|   | ---rw vpls-outgoing-ac* 12vpn-instance-ac-ref |
|   | ---rw vpls-outgoing-pw* 12vpn-instance-pw-ref |
| ---ro entries-count? uint32 |
| ---ro bridge-mrouter-interface* if:interface-ref |
| ---ro vpls-mrouter-interface* 12vpn-instance-pw-ref |
| ---ro group* [address] |
|   | ---ro address inet:ipv6-address |
|   | ---ro mac-address? yang:phys-address |
|   | ---ro expire? uint32 |
|   | ---ro up-time? uint32 |
|   | ---ro last-reporter? inet:ipv6-address |
| ---ro source* [address] |
|   | ---ro address inet:ipv6-address |
|   | ---ro bridge-outgoing-interface* if:interface-ref |
|   | ---ro vpls-outgoing-ac* 12vpn-instance-ac-ref
2.3. IGMP and MLD Snooping References

The IGMP and MLD Snooping instance could be referenced in the scenario of bridge, VPLS to configure the IGMP and MLD Snooping. The name of the instance is the key attribute.

The type of the instance indicates the scenario which is bridge or VPLS. When referenced in bridge, the id of instance means VLAN id. When referenced in VPLS, the id means VSI id.

module: ietf-igmp-mld-snooping
...
+-rw bridges
  | +-rw bridge* [name]
  |    | +-rw name            dot1qtypes:name-type
  |    | +-rw igmp-snooping-instance? igmp-snooping-instance-ref
  |    | +-rw mld-snooping-instance? mld-snooping-instance-ref
  |    | +-rw component* [name]
  |    |   | +-rw name           string
  |    | +-rw bridge-vlan
  |    |   | +-rw vlan* [vid]
  |    |   |    | +-rw vid            dot1qtypes:vlan-index-type
  |    |   | +-rw igmp-snooping-instance? igmp-snooping-instance-ref
  |    |   | +-rw mld-snooping-instance? mld-snooping-instance-ref
  |    |   | +-rw interfaces
  |    |   |   | +-rw interface* [name]
  |    |   |   |    | +-rw name           string
  |    |   |   |    | +-rw igmp-snooping-instance? igmp-snooping-instance-ref

|                    +--rw mld-snooping-instance?    mld-snooping-instance-ref
+-rw l2vpn-instances
    +-rw l2vpn-instance* [name]
      +-rw name                      string
      +-rw igmp-snooping-instance?   igmp-snooping-instance-ref
      +-rw mld-snooping-instance?    mld-snooping-instance-ref
    +-rw endpoint* [name]
      +-rw name                      string
      +-rw igmp-snooping-instance?   igmp-snooping-instance-ref
      +-rw mld-snooping-instance?    mld-snooping-instance-ref
    +--rw (ac-or-pw-or-redundancy-grp)?
       +--:(ac)
          |   +-rw ac* [name]
          |       |   +-rw name                      string
          |       |   +-rw igmp-snooping-instance?   igmp-snooping-instance-ref
          |       |   +-rw mld-snooping-instance?    mld-snooping-instance-ref
          |   +--:(pw)
          |       |   +-rw pw* [name]
          |       |       |   +-rw name                      string
          |       |       |   +-rw igmp-snooping-instance?   igmp-snooping-instance-ref
          |       |       |   +-rw mld-snooping-instance?    mld-snooping-instance-ref
          |   +--:(redundancy-grp)
          |       |   +-rw (primary)
          |       |       |   +-:(primary-ac)
2.4. IGMP and MLD Snooping RPC

IGMP and MLD Snooping RPC clears the specified IGMP and MLD Snooping group tables.

rpcs:
  +---x clear-igmp-snooping-groups {rpc-clear-groups}?  
    |    +---w input  
    |         +---w id?       uint32  
    |         +---w group?    inet:ipv4-address  
    |         +---w source?   inet:ipv4-address  
  +---x clear-mld-snooping-groups {rpc-clear-groups}?  
    +---w input  
         +---w id?       uint32  
         +---w group?    inet:ipv6-address  
         +---w source?   inet:ipv6-address

3. IGMP and MLD Snooping YANG Module

<CODE BEGINS> file "ietf-igmp-mld-snooping@2017-10-25.yang"
module ietf-igmp-mld-snooping {
  // replace with IANA namespace when assigned
  prefix ims;

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

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-l2vpn {
    prefix "l2vpn";
  }

  organization
    "IETF PIM Working Group";

  contact

Zhao
Expires April 26, 2018

[Page 13]
description
"The module defines a collection of YANG definitions common for
IGMP and MLD Snooping."

revision 2017-10-25 {
  description
    "Change model definition to fit NMDA standard.";
  reference
    "RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";
}

revision 2017-08-14 {
  description
    "using profile to cooperate with ieee-dot1Q-bridge module";
  reference
    "RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";
}

revision 2017-06-28 {
  description
    "augment /rt:routing/rt:control-plane-protocols
augment /rt:routing-state/rt:control-plane-protocols";
reference
"RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";
}

revision 2017-02-05 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";
}

/*
  * Features
  */

feature admin-enable {
  description
    "Support configuration to enable or disable IGMP and MLD
    Snooping.";
}

feature fast-leave {
  description
    "Support configuration of fast-leave.";
}

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

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

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

feature per-instance-config {
  description
    "Support configuration of each VLAN or VPLS instance or EVPN
    instance.";
}
feature rpc-clear-groups {
    description
        "Support to clear statistics by RPC for IGMP and MLD Snooping.";
}

feature explicit-tracking {
    description
        "Support configuration of per instance explicit-tracking hosts.";
}

feature exclude-lite {
    description
        "Support configuration of per instance exclude-lite.";
}

/*
 * Typedefs
 */
typedef name-type {
    type string {
        length "0..32";
    }
    description
        "A text string of up to 32 characters, of locally determined significance.";
}
typedef vlan-index-type {
    type uint32 {
        range "1..4094 | 4096..4294967295";
    }
    description
        "A value used to index per-VLAN tables. Values of 0 and 4095 are not permitted. The range of valid VLAN indices. If the value is greater than 4095, then it represents a VLAN with scope local to the particular agent, i.e., one without a global VLAN-ID assigned to it. Such VLANs are outside the scope of IEEE 802.1Q, but it is convenient to be able to manage them in the same way using this YANG module.";
    reference
        "IEEE Std 802.1Q-2014: Virtual Bridged Local Area Networks.";
}
typedef igmp-snooping-instance-ref {
type leafref {
  path "/igmp-snooping-instances/igmp-snooping-instance/name";
}
description "This type is used by data models that need to reference igmp snooping instance.";
}

typedef mld-snooping-instance-ref {
type leafref {
  path "/mld-snooping-instances/mld-snooping-instance/name";
}
description "This type is used by data models that need to reference mld snooping instance.";
}

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

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

typedef source-ipv4-addr-type {
type union {
  type enumeration {
    enum '*' {
      description "Any source address.";
    }
  }
  type inet:ipv4-address;
}
description
typedef source-ipv6-addr-type {
  type union {
    type enumeration {
      enum '*' {
        description "Any source address.";
      }
    }
    type inet:ipv6-address;
  }
  description "Multicast source IP address type.";
} // source-ipv6-addr-type

/*
  * Identities
  */

/*
  * Groupings
  */

grouping general-state-attributes {
  description "Statistics of IGMP and MLD Snooping ";
  container statistics {
    config false;
    description "The statistics of IGMP and MLD Snooping related packets.";
    container received {
      description "Statistics of received messages.";
      uses general-statistics-sent-received;
    }
    container sent {
      description "Statistics of sent messages.";
      uses general-statistics-sent-received;
    }
  } // statistics
} // general-state-attributes
grouping instance-config-attributes-igmp-snooping {
    description "IGMP snooping configuration for each VLAN or VPLS
instance or EVPN instance.";

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

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

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

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

    leaf source-addr {
        type source-ipv4-addr-type;
        description
        "Multicast source IP address.";
    }

    leaf-list bridge-outgoing-interface {
        when "ims:type = 'bridge'";
        type if:interface-ref;
        description "Outgoing interface in bridge fowarding";
    }

    leaf-list vpls-outgoing-ac {
        when "ims:type = 'vpls'";
        type l2vpn-instance-ac-ref;
        description "Outgoing ac in vpls fowarding";
    }
}
leaf-list vpls-outgoing-pw {
    when "ims:type = 'vpls'";
    type 12vpn-instance-pw-ref;
    description "Outgoing pw in vpls forwarding";
}

} // static-l2-multicast-group

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

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

    leaf enable {
        if-feature admin-enable;
        type boolean;
        description "Set the value to true to enable IGMP and MLD Snooping in
the VLAN instance.";
    }

    leaf forwarding-mode {
        type enumeration {
            enum "mac" {
                description "";
            }
            enum "ip" {
                description "";
            }
        }
        description "The default forwarding mode for IGMP and MLD
Snooping is ip.
cisco command is as below
Router(config-vlan-config)# multicast snooping lookup
{ ip | mac } ";
    }

    leaf explicit-tracking {
        if-feature explicit-tracking;
        type boolean;
    }

Zhao                    Expires April 26, 2018                
[Page 20]
description "Tracks IGMP & MLD Snooping v3 membership reports from individual hosts for each port of each VLAN or VSI.";

leaf exclude-lite {
  if-feature exclude-lite;
  type boolean;
  description "lightweight IGMPv3 and MLDv2 protocols, which simplify the standard versions of IGMPv3 and MLDv2.";
  reference "RFC5790";
}

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

/**
leaf mrouter-aging-time {
  type uint16;
  default 180;
  description "Aging time for mrouter interface";
}
/**

leaf fast-leave {
  if-feature fast-leave;
  type empty;
  description "When fast 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 "1..65535";
  }
  units seconds;
  default 1;
  description "Last Member Query Interval, which may be tuned to modify the
leaf query-interval {
    type uint16;
    units seconds;
    default 125;
    description
        "The Query Interval is the interval between General Queries sent by the Querier."
    reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";
}

leaf query-max-response-time {
    type uint16;
    units seconds;
    default 10;
    description
        "Query maximum response time specifies the maximum time allowed before sending a responding report."
    reference "RFC3376. Sec. 4.1.1, 8.3, 8.14.3.";
}

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

leaf robustness-variable {
    type uint8 {
        range "2..7";
    }
    default 2;
    description
        "Querier’s Robustness Variable allows tuning for the expected packet loss on a network."
    reference "RFC3376. Sec. 4.1.6, 8.1, 8.14.1.";
}

leaf version {
    type uint8 {
        range "1..3";
    }
    description "IGMP and MLD Snooping version.";
}

leaf-list static-bridge-mrouter-interface {
    when "ims:type = 'bridge'";
    if-feature static-l2-multicast-group;
    type if:interface-ref;
    description "static mrouter interface in bridge forwarding";
}

leaf-list static-vpls-mrouter-interface {
    when "ims:type = 'vpls'";
    if-feature static-l2-multicast-group;
    type l2vpn-instance-pw-ref;
    description "static mrouter interface in vpls forwarding";
}

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

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

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

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

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

leaf source-addr {
    type source-ipv6-addr-type;
    description
        "Multicast source IP address.";
}

leaf-list bridge-outgoing-interface {
    when "ims:type = 'bridge'";
    type if:interface-ref;
    description "Outgoing interface in bridge forwarding";
}

leaf-list vpls-outgoing-ac {
    when "ims:type = 'vpls'";
    type l2vpn-instance-ac-ref;
    description "Outgoing ac in vpls forwarding";
}

leaf-list vpls-outgoing-pw {
    when "ims:type = 'vpls'";
    type l2vpn-instance-pw-ref;
    description "Outgoing pw in vpls forwarding";
}

} // static-l2-multicast-group

} // instance-config-attributes-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 forwarding."
;
leaf expire {
    type uint32;
    units seconds;
    description "The time left before multicast group timeout.";
}

leaf up-time {
    type uint32;
    units seconds;
    description "The time after the device created L2 multicast record.";
}

} // instance-state-group-attributes-igmp-mld-snooping

grouping instance-state-attributes-igmp-snooping {

description "State attributes for IGMP snooping for each VLAN or VPLS instance or EVPN instance.";

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

list group {
key "address";
config false;

description "IGMP snooping information";

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

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

list source {
  key "address";
  description "Source IP address for multicast stream";
  leaf address {
    type inet:ipv4-address;
    description "Source IP address for multicast stream";
  }
}

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

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

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

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

  leaf host-filter-mode {
    type enumeration {
      enum "include" {
        description "In include mode";
      }
      enum "exclude" {
        description "In exclude mode.";
      }
    }
  }
}
description
"Filter mode for a multicast membership
host may be either include or exclude."
}
} // list host

} // list source
} // list group

// statistics
uses general-state-attributes;

} // instance-state-attributes-igmp-snooping

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

description
"State attributes for both IGMP and MLD Snooping of each
VLAN or VPLS instance or EVPN instance."

leaf entries-count {
    type uint32;
    config false;
    description
        "The number of L2 multicast entries in IGMP and MLD
        Snooping."
}

leaf-list bridge-mrouter-interface {
    when "ims:type = ‘bridge’";
    type if:interface-ref;
    config false;
    description " mrouter interface in bridge fowarding"
}

leaf-list vpls-mrouter-interface {
    when "ims:type = ‘vpls’";
    type l2vpn-instance-pw-ref;
    config false;
    description " mrouter interface in vpls fowarding"
}
grouping instance-state-attributes-mld-snooping {
  description
  "State attributes for MLD snooping of each VLAN.";
  uses instance-state-attributes-igmp-mld-snooping;

  list group {
    key "address";
    config false;
    description "MLD snooping statistics information";
    leaf address {
      type inet:ipv6-address;
      description "Multicast group IP address";
    }
  }
  uses instance-state-group-attributes-igmp-mld-snooping;

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

  list source {
    key "address";
    description "Source IP address for multicast stream";
    leaf address {
      type inet:ipv6-address;
      description "Source IP address for multicast stream";
    }
  }
  uses instance-state-source-attributes-igmp-mld-snooping;

  leaf last-reporter {

type inet:ipv6-address;
description
"The last host address which has sent the report to join
the multicast source and group.";
}

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

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

  leaf host-filter-mode {
    type enumeration {
      enum "include" {
        description
        "In include mode";
      }
      enum "exclude" {
        description
        "In exclude mode.";
      }
    }
    description
    "Filter mode for a multicast membership
    host may be either include or exclude.";
  }
}

// list host

} // list source
} // list group

// statistics
uses general-state-attributes;

} // instance-state-attributes-mld-snooping

grouping instance-state-source-attributes-igmp-mld-snooping {
  description
  "State attributes for both IGMP and MLD Snooping of each VLAN
  or VPLS instance or EVPN instance.";

leaf-list bridge-outgoing-interface {
  when "ims:type = 'bridge'";
  type if:interface-ref;
  description "Outgoing interface in bridge forwarding";
}

leaf-list vpls-outgoing-ac {
  when "ims:type = 'vpls'";
  type l2vpn-instance-ac-ref;
  description "Outgoing ac in vpls forwarding";
}

leaf-list vpls-outgoing-pw {
  when "ims:type = 'vpls'";
  type l2vpn-instance-pw-ref;
  description "Outgoing pw in vpls forwarding";
}

leaf up-time {
  type uint32;
  units seconds;
  description "The time after the device created L2 multicast record";
}

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

leaf host-count {
  if-feature explicit-tracking;
  type uint32;
  description "The number of host addresses.";
}

} // instance-state-source-attributes-igmp-mld-snooping

grouping general-statistics-error {
  description
"A grouping defining statistics attributes for errors."

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.";
}

// general-statistics-error

grouping general-statistics-sent-received {
    description
    "A grouping defining statistics attributes.";

    leaf query {
        type yang:counter64;
        description
        "The number of query messages.";
    }
    leaf membership-report-v1 {
        type yang:counter64;
        description
        "The number of membership report v1 messages.";
    }
    leaf membership-report-v2 {
        type yang:counter64;
        description
        "The number of membership report v2 messages.";
    }
    leaf membership-report-v3 {
        type yang:counter64;
        description
        "The number of membership report v3 messages.";
    }
    leaf leave {
        type yang:counter64;
        description
        "The number of leave messages.";
    }
    leaf pim {
        type yang:counter64;
        description
        "The number of pim hello messages.";
    }
grouping endpoint-grp {
  description "A grouping that defines the structure of " +
  "an endpoint";
  choice ac-or-pw-or-redundancy-grp {
    description "A choice of attachment circuit or " +
    "pseudowire or redundancy group";
    case ac {
      description "Attachment circuit(s) as an endpoint";
      list ac {
        key "name";
        leaf name {
          type string;
          description "Name of attachment circuit. " +
          "This field is intended to " +
          "reference standardized " +
          "layer-2 definitions.";
        }
        leaf igmp-snooping-instance {
          type igmp-snooping-instance-ref;
          description "Configure igmp-snooping instance under" +
          "the bridge view";
        }
        leaf mld-snooping-instance {
          type mld-snooping-instance-ref;
          description "Configure mld-snooping instance under the" +
          "bridge view";
        }
      }
    }
    case pw {
      description "Pseudowire(s) as an endpoint";
      list pw {
        key "name";
        leaf name {
          type string;
          description "Name of Pseudowire."
        }
        leaf igmp-snooping-instance {
          type igmp-snooping-instance-ref;
        }
      }
    }
    case redundancy {
      description "Redundancy group(s) as an endpoint";
      list redundancy {
        key "name";
        leaf name {
          type string;
          description "Name of redundancy group."
        }
        leaf igmp-snooping-instance {
          type igmp-snooping-instance-ref;
        }
      }
    }
  }
}

// general-statistics-sent-received
description "Configure igmp-snooping instance under the bridge view";
}
leaf mld-snooping-instance {
  type mld-snooping-instance-ref;
  description "Configure mld-snooping instance under the bridge view";
}

description "An L2VPN instance's " + "pseudowire(s) list";
}
}
case redundancy-grp {
  description "Redundancy group as an endpoint";
  choice primary {
    mandatory true;
    description "primary options";
    case primary-ac {
      description "primary-ac"
      container primary-ac {
        description "Primary AC"
        leaf name {
          type string;
          description "Name of attachment circuit. ";
        }
      }
      leaf igmp-snooping-instance {
        type igmp-snooping-instance-ref;
        description "Configure igmp-snooping instance under the bridge view";
      }
      leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping instance under the bridge view";
      }
    } // primary-ac
  } // primary-ac

case primary-pw {
  list primary-pw {
    key "name";
    leaf name {
      type string;
      description "Name of Pseudowire.";
    }
  }
}
leaf igmp-snooping-instance {
    type igmp-snooping-instance-ref;
    description "Configure igmp-snooping instance under the bridge view";
}
leaf mld-snooping-instance {
    type mld-snooping-instance-ref;
    description "Configure mld-snooping instance under the bridge view";
}

description "primary-pw";

} //primary-pw
} //primary-pw

choice backup {
    description "backup options";
    case backup-ac {
        description "backup-ac";
        container backup-ac {
            description "Backup AC";
            leaf name {
                type string;
                description "Name of attachment circuit.";
            }
            leaf igmp-snooping-instance {
                type igmp-snooping-instance-ref;
                description "Configure igmp-snooping instance under the bridge view";
            }
            leaf mld-snooping-instance {
                type mld-snooping-instance-ref;
                description "Configure mld-snooping instance under the bridge view";
            }
        } // backup-ac
    } // backup-ac
    case backup-pw {
        description "backup-pw";
        list backup-pw {
            key "name";
            leaf name {
                type string;
                description "Name of Pseudowire."
            }
            leaf igmp-snooping-instance {
                type igmp-snooping-instance-ref;
                description "Configure igmp-snooping instance under the bridge view";
            }
            leaf mld-snooping-instance {
                type mld-snooping-instance-ref;
                description "Configure mld-snooping instance under the bridge view";
            }
        } // backup-pw
    } // backup-pw
} // backup-pw
} // backup-pw
type igmp-snooping-instance-ref;

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

description "backup-pw";
} //backup-pw

/*
 * igmp-snooping-instance
 */
container igmp-snooping-instances {
  description
  "igmp-snooping-instance list";

  list igmp-snooping-instance {
    key "name";
    description
      "IGMP Snooping instance to configure the igmp-
snooping.";

    leaf name {
      type string;
      description
        "Name of the igmp-snooping-instance to configure the igmp
snooping.";
    }

    leaf id {
      type uint32;
      description
        "It is vlan_id or vpls_id.
When igmp-snooping-instance is applied under bridge view, its
value is 0.";
    }

Zhao                    Expires April 26, 2018                
[Page 35]
leaf type {
    type enumeration {
        enum "bridge" {
            description "bridge";
        }
        enum "vpls" {
            description "vpls";
        }
    }
    description "The type indicates bridge or vpls.";
}

uses instance-config-attributes-igmp-snooping {
    if-feature per-instance-config;
}

uses instance-state-attributes-igmp-snooping;

} //igmp-snooping-instance
} //igmp-snooping-instances

/*
 * mld-snooping-instance
 */
container mld-snooping-instances {
    description "mld-snooping-instance list";

list mld-snooping-instance {
    key "name";
    description "MLD Snooping instance to configure the mld-snooping.";

    leaf name {
        type string;
        description "Name of the mld-snooping-instance to configure the mld
snooping.";
    }

    leaf id {


```yam

leaf type {
  type enumeration {
    enum "bridge" {
      description "bridge";
    }
    enum "vpls" {
      description "vpls";
    }
  }
  description "The type indicates bridge or vpls.";
}

uses instance-config-attributes-mld-snooping {
  if-feature per-instance-config;
}

uses instance-state-attributes-mld-snooping;

} //mld-snooping-instance
} //mld-snooping-instances

container bridges {
  description "Apply igmp-mld-snooping instance in the bridge scenario";
  list bridge {
    key name;
    description "bridge list";
    leaf name {
      type name-type;
      description "bridge name";
    }
  }
}
```
leaf igmp-snooping-instance {
    type igmp-snooping-instance-ref;
    description "Configure igmp-snooping instance under the
bridge view";
}
leaf mld-snooping-instance {
    type mld-snooping-instance-ref;
    description "Configure mld-snooping instance under the
bridge view";
}
list component {
    key "name";
    description "";

    leaf name {
        type string;
        description "The name of the Component.";
    }
    container bridge-vlan {
        description "bridge vlan";
        list vlan {
            key "vid";
            description "";

            leaf vid {
                type vlan-index-type;
                description "The VLAN identifier to which this entry
applies.";
            }
        }
    }
    leaf igmp-snooping-instance {
        type igmp-snooping-instance-ref;
        description "Configure igmp-snooping instance
under the vlan view";
    }
    leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping instance
under the vlan view";
    }
    container interfaces {
        description "Interface configuration parameters.";
    }

list interface {
    key "name";
    description
        "The list of configured interfaces on the device.";

    leaf name {
        type string;
        description
            "The name of the interface.";
    }

    leaf igmp-snooping-instance {
        type igmp-snooping-instance-ref;
        description "Configure igmp-snooping instance under the interface view";
    }

    leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping instance under the interface view";
    }
}

container l2vpn-instances {
    description "Apply igmp-mld-snooping instance in the vpls scenario";

    list l2vpn-instance {
        key "name";
        description "An VPLS service instance";

        leaf name {
            type string;
            description "Name of VPLS service instance";
        }
    }
}
leaf igmp-snooping-instance {
    type igmp-snooping-instance-ref;
    description "Configure igmp-snooping instance under the l2vpn-instance view";
}
leaf mld-snooping-instance {
    type mld-snooping-instance-ref;
    description "Configure mld-snooping instance under the l2vpn-instance view";
}

list endpoint {
    key "name";
    description "An endpoint";
    leaf name {
        type string;
        description "endpoint name";
    }
    leaf igmp-snooping-instance {
        type igmp-snooping-instance-ref;
        description "Configure igmp-snooping instance under the interface view";
    }
    leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping instance under the interface view";
    }
}

uses endpoint-grp;

} //endpoint

/*
* RPCs
*/

rpc clear-igmp-snooping-groups {
    if-feature rpc-clear-groups;
    description "Clears the specified IGMP Snooping cache tables.";
    input {
        leaf id {
            type uint32;
        }
    }
}
description
"VLAN ID, VPLS ID, or EVVPN ID";
}

leaf group {
  type inet:ipv4-address;
  description
  "Multicast group IPv4 address.
  If it is not specified, all IGMP snooping group tables
  are
  cleared.";
}

leaf source {
  type inet:ipv4-address;
  description
  "Multicast source IPv4 address.
  If it is not specified, all IGMP snooping source-group
  tables are
  cleared.";
}
}
} // rpc clear-igmp-snooping-groups

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

  input {
    leaf id {
      type uint32;
      description
      "VLAN ID, VPLS ID, or EVVPN ID";
    }

    leaf group {
      type inet:ipv6-address;
      description
      "Multicast group IPv6 address.
      If it is not specified, all MLD snooping group tables are
      cleared.";
    }

    leaf source {
      type inet:ipv6-address;
      description
      "Multicast source IPv6 address.
      If it is not specified, all MLD snooping source-group
      tables are
      cleared.";
    }
  }
}
"Multicast source IPv6 address.
If it is not specified, all MLD snooping source-group
tables are
cleared.";
}
}
} // rpc clear-mld-snooping-groups

4. Security Considerations

The data model defined does not create any security implications.

5. IANA Considerations

This draft does not request any IANA action.

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
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA

EMail: Xufeng_Liu@jabil.com

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

Email: liuyisong@huawei.com

Anish Peter
Individual

EMail: anish.ietf@gmail.com

Mahesh Sivakumar
Cisco Systems
510 McCarthy Boulevard
Milpitas, California
USA