Abstract

PIM is widely deployed multicast protocol. PIM protocol is defined in [RFC4601] and [RFC7761]. As deployment for PIM protocol growing day by day, user expect least traffic loss and fast convergence in case of any network failure. This document provides extension to existing defined protocol which would improve stability of PIM protocol with respect to traffic loss and convergence time when the PIM DR is down.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 8, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction .................................................. 2
2. Terminology ..................................................... 3
3. PIM hello message format ...................................... 3
   3.1. DR Address Option format .................................. 4
   3.2. BDR Address Option format ................................. 4
4. The Protocol Treatment .......................................... 4
   4.1. Election Algorithm ......................................... 5
   4.2. Sending Hello Messages .................................... 6
   4.3. Receiving Hello Messages .................................. 7
   4.4. The treatment ................................................ 8
   4.5. Sender side .................................................. 9
5. Compatibility ..................................................... 9
6. Deployment suggestion ........................................... 9
7. Security Considerations .......................................... 9
8. IANA Considerations ............................................ 10
9. Normative References ........................................... 10
Authors’ Addresses ............................................... 10

1. Introduction

Multicast technology is used widely. Many modern technology use PIM technology, such as IPTV, Net-Meeting, and so on. There are many events that will influence the quality of multicast services. The change of unicast routes will cause the lost of multicast packets. The change of DR cause the lost of multicast packets too.

When a DR on a share-media LAN is down, other routers will elect a new DR until the expiration of Hello-Holdtime. The default value of Hello-Holdtime is 105 seconds. Although the value of Hello-Holdtime can be changed by manual, when the DR is down, there are still many multicast packets will be lost. The quality of IPTV and Net-Meeting will be influenced.
For example, there were two routers on one Ethernet. RouterA was elected to DR. When RouterA is down, the multicast packets are discarded until the RouterB is elected to DR and RouterB imports the multicast flows successfully.

We suppose that there is only a RouterA in the Ethernet at first in Figure 1. RouterA is the DR which is responsible for forwarding multicast flows. When RouterB connects the Ethernet, RouterB will be elected to DR because a higher priority. So RouterA will stop forwarding multicast packets. The multicast flows will not recover until RouterB joins the multicast group after it is elected to DR.

2. Terminology

Backup Designated Router (BDR): A shared-media LAN like Ethernet may have multiple PIM-SM routers connected to it. Except for DR, a router which will act on behalf of directly connected hosts with respect to the PIM-SM protocol. But BDR will not forward the flows. When DR is down, the BDR will forward multicast flows immediately. A single BDR is elected per interface like the DR.

3. PIM hello message format

In [RFC4601] and [RFC7761], the PIM hello message format was defined. In this document, we define two new option values which are including Type, Length, and Value.
3.1. DR Address Option format

- **OptionType**: The value is TBD.
- **OptionLength**: If the network is support IPv4, the OptionLength is 4 octets. If the network is support IPv6, the OptionLength is 16 octets.
- **OptionValue**: The OptionValue is IP address of DR. If the network is support IPv4, the value is IPv4 address of DR. If the network is support IPv6, the value is IPv6 address of DR.

3.2. BDR Address Option format

- **OptionType**: The value is TBD.
- **OptionLength**: If the network is support IPv4, the OptionLength is 4 octets. If the network is support IPv6, the OptionLength is 16 octets.
- **OptionValue**: The OptionValue is IP address of BDR. If the network is support IPv4, the value is IPv4 address of BDR. If the network is support IPv6, the value is IPv6 address of BDR.

4. The Protocol Treatment

A new router starts to send hello messages with the values of DR and BDR are all set to 0 after its interface is enabled in PIM on a share-media LAN. When the router receives hello messages from other routers on the same share-media LAN, the router will check if the value of DR is filled. If the value of DR is filled with IP address of router which is sending hello messages, the router will store the IP address as the DR address of this interface.
Then the new router compare the priority and IP address itself to the stored IP address of DR and BDR according to the algorithm of [RFC4601] and [RFC7761]. If the new router notices that it is better to be DR than the existed DR or BDR. The router will make itself the BDR, and send new hello messages with its IP address as BDR and existed DR. If the router notices that the existed DR has the highest priority in the share-media LAN, but the existed BDR is set to 0x0 in the received hello messages, or the existed BDR is not better than the new router, the new router will elect itself to BDR. If the router notices that it is not better to be DR than existed DR and BDR, the router will respect the existed DR and BDR.

When the new router becomes the new BDR, the router will join the existed multicast groups, import multicast flows from upstream routers. But the BDR MUST not forward the multicast flows to avoid the duplicate multicast packets in the share-media LAN. The new router will monitor the DR. The method that BDR monitors the DR may be BFD technology or other ways that can be used to detect link/node failure quickly. When the DR becomes unavailable because of the down or other reasons, the BDR will forward multicast flows immediately.

DR / BDR election SHOULD be handled in two ways. Selection of which procedure to use would be totally dependent on deployment scenario.

1. When new router is added in existing steady state of network, if the new router notices that it is better to be DR than the existing DR. It does elect itself as DR as procedure defined in RFC 7761. This method must be used when user is ok to have transition in network. This option should be used if deployment requirement is to adopt with new DR as and when they are available, and intermediate network transition is acceptable.

2. If the new router notices that it is better to be DR than the existed DR or BDR, the router will make itself the BDR, and send new hello message with its IP address as BDR and existed DR. Uses of this option would have less transition in network. This option should be used when deployment requirement is to have minimum transition in network unless there is some failure.

4.1. Election Algorithm

The DR and BDR election is according the rules defined below, the algorithm is similar to the DR election definition in [RFC2328].

(1) Note the current values for the network’s Designated Router and Backup Designated Router. This is used later for comparison purposes.
(2) Calculate the new Backup Designated Router for the network as follows. Those routers that have not declared themselves to be Designated Router are eligible to become Backup Designated Router. The one which have the highest priority will be chosen to be Backup Designated Router. In case of a tie, the one having the highest Router ID is chosen.

(3) Calculate the new Designated Router for the network as follows. If one or more of the routers have declared themselves Designated Router (i.e., they are currently listing themselves as Designated Router in their Hello Packets) the one having highest Router Priority is declared to be Designated Router. In case of a tie, the one having the highest Router ID is chosen. If no routers have declared themselves Designated Router, assign the Designated Router to be the same as the newly elected Backup Designated Router.

(4) If Router X is now newly the Designated Router or newly the Backup Designated Router, or is now no longer the Designated Router or no longer the Backup Designated Router, repeat steps 2 and 3, and then proceed to step 5. For example, if Router X is now the Designated Router, when step 2 is repeated X will no longer be eligible for Backup Designated Router election. Among other things, this will ensure that no router will declare itself both Backup Designated Router and Designated Router.

(5) As a result of these calculations, the router itself may now be Designated Router or Backup Designated Router.

The reason behind the election algorithm’s complexity is the desire for the DR stability.

The above procedure may elect the same router to be both Designated Router and Backup Designated Router, although that router will never be the calculating router (Router X) itself. The elected Designated Router may not be the router having the highest Router Priority. If Router X is not itself eligible to become Designated Router, it is possible that neither a Backup Designated Router nor a Designated Router will be selected in the above procedure. Note also that if Router X is the only attached router that is eligible to become Designated Router, it will select itself as Designated Router and there will be no Backup Designated Router for the network.

4.2. Sending Hello Messages

According to Section 4.3.1 in [RFC4601] and [RFC7761], when a new router’s interface is enabled in PIM protocol, the router send hello messages with the values of DR and BDR are filled with 0x0. Then the interface is in waiting state and start the hold-timer which is like
the neighbor hold-timer. When the hold-timer is expired, the interface will elect the DR and BDR according to the DR election rules.

When a new router sets itself BDR after receiving hello messages from other routers, the router sends hello messages with the value of DR set to the IP address of the existed DR and the value of BDR is set to the IP address of the router itself.

When a existed BDR sets itself non DR and non BDR after receiving hello messages from other routers, the router will send hello messages with the value of DR is set to existed DR and the value of BDR is set to new BDR.

```
DR \       newcomer /
   ├──       └────        ┌──
   | A |       | B |           | C |
   └──       └──           └──
                   ┌──
                   |           |
                   |           |
                   │           │
                   │           │
------------------------ LAN
```

For example, there is a stable LAN that include RouterA and RouterB. RouterA is the DR which has the best priority. RouterC is a newcomer. RouterC sends hello packet with the DR and BDR is all set to zero.

If RouterC cannot send hello packet with the DR/BDR capability, Router C may send the hello packet according the rule defined in [RFC4601] and [RFC7761].

If deployment requirement is to adopt with new DR as and when they are available, a new router with highest priority or best IP address sends hello packet with DR and BDR all set to zero at first. It sends hello packet with itself set to DR after it finish join all the existing multicast groups. Then existed DR compares with the new router, the new router will be final DR.

4.3. Receiving Hello Messages

When the values of DR and BDR which are carried by hello messages are received is all set to 0x0, the router MUST elect the DR using procedure defined in [RFC4601] and [RFC7761] after the hold-timer expires. And elect a new BDR which is the best choice except DR.
In case the value of DR which is carried by received hello messages is not 0x0, and the value of BDR is set to 0x0, when the hold-timer expires there is no hello packet from other router is received, the router will elect itself to BDR.

In case either of the values of DR and BDR that are carried by received hello messages are larger than 0x0. The router will mark the existed DR, and compare itself and the BDR in message. When the router notice that it is better to be DR than existed BDR. The router will elect itself to the BDR.

When a router receives a new hello message with the values of DR and BDR are set to 0x0. The router will compare the new router with existed information. If the router noticed that the new router is better to be DR than itself, or the new router is better to be BDR than the existed BDR, the router will set the BDR to the new router.

When existed DR receives hello packet with DR set larger than zero, algorithm defined in section 4.1 can be used to select the final DR.

As illustrated in Figure 3, after RouterC sends hello packet, RouterC will not elect the DR until hold-timer expired. During the period, RouterC should receive the hello packets from RouterA and RouterB. RouterC accepts the result that RouterA is the DR. In case RouterC has the lowest priority than RouterA and RouterB, RouterC will also accept that Router B is the BDR. In case RouterC has the intermediate priority among the three routers, RouterC will treat itself as new BDR after the hold-timer expired. In case RouterC has the highest priority among the three routers, RouterC will treat RouterA which is the existed DR as DR, and RouterC will treat itself as new BDR. If the network administrator thinks that RouterC should be new DR, the DR changing should be triggered manually.

Exception: During the hold-timer period, RouterC receives only the hello packet from RouterA. When the hold-timer expired, RouterC treats RouterA as DR. and RouterC treats itself as BDR. In case RouterC only receives the hello packet from RouterB during the hold-timer period, RouterC will compare the priority between RouterB and itself to elect the new DR. In these situations, some interfaces or links go wrong in the LAN.

4.4. The treatment

When all the routers on a shared-media LAN are start to work on the same time, the election result of DR is same as [RFC4601] and [RFC7761]. And all the routers will elect a BDR which is suboptimum to DR. The routers in the network will store the DR and BDR. The
hello messages sent by all the routers are same with the value of DR and BDR are all set.

When a new router starts to work on a shared-media LAN and receive hello messages from other routers that the value of DR is set. The new router will not change the existed DR even if it is superior to the existed DR. If the new router is superior to existed BDR, the new router will replace the existed BDR.

When the routers receive hello message from a new router, the routers will compare the new router and all the other routers on the LAN. If the new router is superior to existed BDR, the new router will be new BDR. Then the old BDR will send prune message to upstream routers.

As a result, the BDR is the one which has the highest priority except DR. Once the DR is elected, the DR will not change until it fails or manually adjustment. After the DR and BDR are elected, the routers in the network will store the address of DR and BDR.

4.5. Sender side

DR/BDR function also can be used in source side that multiple routers and source is in same share-media network. The algorithm is the same as the receiver side. Only the BDR need not build multicast tree from downstream router.

5. Compatibility

If the LAN is a hybrid network that there are some routers which have DR/BDR capability and the other routers which have not DR/BDR capability. In order to avoid duplicated multicast flows in the LAN, all the routers should go backward to use the algorithm defined in [RFC4601] and [RFC7761].

6. Deployment suggestion

If there are two and more routers on a share-media LAN, and the multicast services is sensitive to the lost of multicast packets, the function of DR and BDR defined in this document should be deployed.

7. Security Considerations

For general PIM Security Considerations.
8. IANA Considerations

IANA is requested to allocate OptionTypes in TLVs of hello message. Include DR and BDR.

9. Normative References


Authors' Addresses

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

Email: zhang.zheng@zte.com.cn
Abstract

PIM is widely deployed multicast protocol. PIM protocol is defined in [RFC7761]. As deployment for PIM protocol is growing day by day, user expects lower traffic loss and faster convergence in case of any network failure. This document provides extension to the existing protocol which would improve stability of PIM protocol with respect to traffic loss and convergence time when the PIM DR is down.

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 December 30, 2018.

Copyright Notice

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

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

Multicast technology is used widely. Many modern technologies, such as IPTV, Net-Meeting, use PIM-SM to facilitate multicast service. There are many events that will influence the quality of multicast services. Like the change of unicast routes, the change of the PIM-SM DR may cause the loss of multicast packets too.

After a DR on a shared-media LAN went down, other routers will elect a new DR after the expiration of Hello-Holdtime. The default value of Hello-Holdtime is 105 seconds. Although the minimum Hello interval can be adjust to 1 second and the Hello-Holdtime is 3.5 times Hello interval. Thus, the detection of DR Down event cannot be guaranteed in less than 3.5 seconds. And it is still too long for modern multicast services. Still, may multicast packets will be lost. The quality of IPTV and Net-Meeting will be influenced.
For example, there are two routers on one Ethernet. RouterA is elected to DR. When RouterA is down, the multicast packets are discarded until the RouterB is elected to DR and RouterB imports the multicast flows successfully.

We suppose that there is only a RouterA in the Ethernet at first in Figure 1. RouterA is the DR which is responsible for forwarding multicast flows. When RouterB connects to the Ethernet segment, RouterB will be elected as DR because of its higher priority. So RouterA will stop forwarding multicast packets. The multicast flows will not recover until RouterB pulls the multicast flows after it is elected to DR.

So if we want to increase the stability of DR, carrying DR/ BDR role information in PIM hello packet is a feasible way to show the DR/ BDR roles explicitly. It avoids the confusion caused by new comers which has a higher priority.

2. Terminology

Backup Designated Router (BDR): Like DR, A BDR which acts on behalf of directly connected hosts in a shared-media LAN. But BDR MUST not forward the flows when DR works normally. When DR is down, the BDR will forward multicast flows immediately. A single BDR MUST be elected per interface like the DR.

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

3. PIM hello message format

In [RFC7761], the PIM hello message format is defined. In this document, we define two new option values which are including Type, Length, and Value.
3.1. DR Address Option format

- OptionType: The value is TBD.
- OptionLength: If the network is support IPv4, the OptionLength is 4 octets. If the network is support IPv6, the OptionLength is 16 octets.
- OptionValue: The OptionValue is IP address of DR. If the network is support IPv4, the value is IPv4 address of DR. If the network is support IPv6, the value is IPv6 address of DR.

3.2. BDR Address Option format

- OptionType: The value is TBD.
- OptionLength: If the network is support IPv4, the OptionLength is 4 octets. If the network is support IPv6, the OptionLength is 16 octets.
- OptionValue: The OptionValue is IP address of BDR. If the network is support IPv4, the value is IPv4 address of BDR. If the network is support IPv6, the value is IPv6 address of BDR.

4. The Protocol Treatment

Carrying DR/ BDR role information in PIM hello packet is a feasible way to keep the stability of DR. It avoid the confusion caused by new comers which has a higher priority. So there are some changes in PIM hello procedure and interface state machine.

A new router starts to send hello messages with the values of DR and BDR are all set to 0 after its interface is enabled in PIM on a shared-media LAN. When the router receives hello messages from other routers on the same shared-media LAN, the router will check if the
value of DR is filled. If the value of DR is filled with IP address of router which is sending hello messages, the router will store the IP address as the DR address of this interface.

Then the new router compares the priority and IP address itself to the stored information of DR and BDR according to the algorithm of [RFC7761]. If the new router notices that it is better to be DR than the current DR or BDR. The router will make itself the BDR, and send new hello messages with its IP address as BDR and current DR. If the router notices that the current DR has the highest priority in the shared-media LAN, but the current BDR is set to 0x00000000 if IPv4 addresses are in use or 0:0:0:0:0:0:0:0/128 if IPv6 addresses are in use in the received hello messages (To be simplify, we use 0x0 in abbreviation in following parts of the draft), or the current BDR is not better than the new router, the new router will elect itself to BDR. If the router notices that it is not better to be DR than current DR and BDR, the router will follow the current DR and BDR.

When the new router becomes the new BDR, the router will join the current multicast groups, import multicast flows from upstream routers. But the BDR MUST not forward the multicast flows to avoid the duplicate multicast packets in the shared-media LAN. The new router will monitor the DR. The method that BDR monitors the DR may be BFD technology or other ways that can be used to detect link/node failure quickly. When the DR becomes unavailable because of the down or other reasons, the BDR will forward multicast flows immediately.

4.1. Deployment Choice

DR / BDR election SHOULD be handled in two ways. Selection of which procedure to use would be totally dependent on deployment scenario.

1. The algorithm defined in [RFC7761] should be used if it is ok to adopt with new DR as and when they are available, and the loss caused by DR changing is acceptable.

2. If the deployment requirement is to have minimum packets loss when DR changing the mechanism defined in this draft should be used. That is, if the new router notices that it is better to be DR than the current DR or BDR, the router will make itself the BDR, and send new hello message with its IP address as BDR and current DR.

According to section 4.9.2 defined in [RFC7761], the device receives unknown options Hello packet will ignore it. So the new extension defined in this draft will not influence the stability of neighbor. But if the router which has the ability defined in this draft receives non-DR/BDR capable Hello messages defined in [RFC7761], the router MAY stop sending DR/BDR capable Hello messages in the LAN and
4.2. Election Algorithm

The DR and BDR election is according the rules defined below, the algorithm is similar to the DR election definition in [RFC2328].

(1) Note the current values for the network’s Designated Router and Backup Designated Router. This is used later for comparison purposes.

(2) Calculate the new Backup Designated Router for the network as follows. The router that has not declared itself to be Designated Router is eligible to become Backup Designated Router. The one which have the highest priority will be chosen to be Backup Designated Router. In case of a tie, the one having the highest Router ID is chosen.

(3) Calculate the new Designated Router for the network as follows. If one or more of the routers have declared themselves Designated Router (i.e., they are currently listing themselves as Designated Router in their Hello Packets) the one having highest Router Priority is declared to be Designated Router. In case of a tie, the one having the highest Router ID is chosen. If no routers have declared themselves Designated Router, assign the Designated Router to be the same as the newly elected Backup Designated Router.

(4) If Router X is now newly the Designated Router or newly the Backup Designated Router, or is now no longer the Designated Router or no longer the Backup Designated Router, repeat steps 2 and 3, and then proceed to step 5. For example, if Router X is now the Designated Router, when step 2 is repeated X will no longer be eligible for Backup Designated Router election. Among other things, this will ensure that no router will declare itself both Backup Designated Router and Designated Router.

(5) As a result of these calculations, the router itself may now be Designated Router or Backup Designated Router.

The reason behind the election algorithm’s complexity is the desire for the DR stability.

The above procedure may elect the same router to be both Designated Router and Backup Designated Router, although that router will never be the calculating router (Router X) itself. The elected Designated Router may not be the router having the highest Router Priority. If Router X is not itself eligible to become Designated Router, it is
possible that neither a Backup Designated Router nor a Designated Router will be selected in the above procedure. Note also that if Router X is the only attached router that is eligible to become Designated Router, it will select itself as Designated Router and there will be no Backup Designated Router for the network.

4.3. Sending Hello Messages

According to Section 4.3.1 in [RFC7761], when a new router’s interface is enabled in PIM protocol, the router sends Hello messages with the values of DR and BDR are filled with 0x0. Then the interface is in Waiting state and start the hold-timer which is equal to the Neighbor Liveness Timer. When the timer is expired, the interface will elect the DR and BDR according to the DR election rules.

When a new router sets itself BDR after receive hello messages from other routers, the router send hello messages with the value of DR is set to the IP address of current DR and the value of BDR is set to the IP address of the router itself.

A current BDR MUST set itself DROther after it receives Hello messages from other routers, the router will send hello messages with the value of DR is set to current DR and the value of BDR is set to new BDR.

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

Figure 3

For example, there is a stable LAN that includes RouterA and RouterB. RouterA is the DR which has the best priority. RouterC is a newcomer. RouterC sends hello packet with the DR and BDR is all set to zero.

If RouterC cannot send hello packet with the DR/BDR capability, Router C MAY send the hello packet according the rule defined in [RFC7761].

If deployment requirement is to adopt with new DR as and when they are available, a new router with highest priority or best IP address sends hello packet with DR and BDR all set to zero at first. It
sends hello packet with itself set to DR after it finish join all the existing multicast groups. Then current DR compares with the new router, the new router will be final DR.

4.4. Receiving Hello Messages

When the values of DR and BDR which are carried by hello messages are received is all set to 0x0, the router MUST elect the DR using procedure defined in DR election algorithm after the hold-timer expires. And elect a new BDR which is the best choice except DR. The election cases can be executed as following:

In case the value of DR which is carried by received hello messages is not 0x0, and the value of BDR is set to 0x0, when the hold-timer expires there is no hello packet from other router is received, the router will elect itself to BDR.

In case either of the values of DR and BDR that are carried by received hello messages are larger than 0x0. The router will mark the current DR, and compare itself and the BDR in message. When the router notice that it is better to be DR than current BDR. The router will elect itself to the BDR.

When a router receives a new hello message with the values of DR and BDR are set to 0x0. The router will compare the new router with current information. If the router noticed that the new router is better to be DR than itself, or the new router is better to be BDR than the current BDR, the router will set the BDR to the new router.

When current DR receives hello packet with DR set larger than zero, algorithm defined in section 4.1 can be used to select the final DR.

As illustrated in Figure 3, after RouterC sends hello packet, RouterC will not elect the DR until hold-timer expired. During the period, RouterC should receive the hello packets from RouterA and RouterB. RouterC accepts the result that RouterA is the DR. In case RouterC has the lowest priority than RouterA and RouterB, RouterC will also accept that Router B is the BDR. In case RouterC has the intermediate priority among the three routers, RouterC will treat itself as new BDR after the hold-timer expired. In case RouterC has the highest priority among the three routers, RouterC will treat RouterA which is the current DR as DR, and RouterC will treat itself as new BDR. If the network administrator thinks that RouterC should be new DR, the DR changing should be triggered manually.

Exception: During the hold-timer period, RouterC receives only the hello packet from RouterA. When the hold-timer expired, RouterC treats RouterA as DR. and RouterC treats itself as BDR. In case
RouterC only receives the hello packet from RouterB during the hold-timer period, RouterC will compare the priority between RouterB and itself to elect the new DR. In these situations, some interfaces or links go wrong in the LAN.

4.5. The treatment

When all the routers on a shared-media LAN are start to work on the same time, the election result of DR is same as [RFC7761]. And all the routers will elect a BDR which is next best to DR. The routers in the network will store the DR and BDR. The hello messages sent by all the routers are same with the value of DR and BDR are all set.

When a new router start to work on a shared-media LAN and receive hello messages from other routers that the value of DR is set. The new router will not change the current DR even if it is superior to the current DR. If the new router is superior to current BDR, the new router will replace the current BDR.

When the routers receive hello message from a new router, the routers will compare the new router and all the other routers on the LAN. If the new router is superior to current BDR, the new router will be new BDR. Then the old BDR will send prune message to upstream routers.

As a result, the BDR is the one which has the highest priority except DR. Once the DR is elected, the DR will not change until it fails or manually adjustment. After the DR and BDR are elected, the routers in the network will store the address of DR and BDR.

4.6. Sender side

DR/BDR function also can be used in source side that multiple routers and source is in same shared-media network. The algorithm is the same as the receiver side. Only the BDR need not build multicast tree from downstream router.

5. Compatibility

If the LAN is a hybrid network that there are some routers which have DR/BDR capability and the other routers which have not DR/BDR capability. All the routers MAY go backward to use the algorithm defined in [RFC7761].

6. Deployment suggestion

If there are two and more routers which is responsible for multicast flow forwarding on a shared-media LAN, and the multicast services is
sensitive to the lost of multicast packets, the function of DR and BDR defined in this document SHOULD be deployed.

7. Security Considerations

For general PIM Security Considerations.

8. IANA Considerations

IANA is requested to allocate OptionType in TLVs of hello message. Include DR and BDR.

9. Acknowledgements

The authors would like to thank Greg Mirsky for their valuable comments and suggestions.

10. Normative References


Authors' Addresses

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

Email: zhang.zheng@zte.com.cn
Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai
China
Email: hu.fangwei@zte.com.cn

Benchong Xu
ZTE Corporation
No. 68 Zijinghua Road, Yuhuatai Distinct
Nanjing
China
Email: xu.benchong@zte.com.cn

Mankamana Mishra
Cisco Systems
821 Alder Drive,
MILPITAS, CALIFORNIA 95035
UNITED STATES
Email: mankamis@cisco.com
PIM Designated Router Load Balancing
draft-ietf-pim-drlb-06

Abstract

On a multi-access network, one of the PIM routers is elected as a Designated Router (DR). On the last hop LAN, the PIM DR is responsible for tracking local multicast listeners and forwarding traffic to these listeners if the group is operating in PIM-SM. In this document, we propose a modification to the PIM-SM protocol that allows more than one of these last hop routers to be selected so that the forwarding load can be distributed among these routers.

Status of This Memo

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

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

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

This Internet-Draft will expire on December 30, 2017.

Copyright Notice

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

1.  Introduction .................................................  2
2.  Terminology ..................................................  5
3.  Applicability .................................................  5
4.  Functional Overview ..........................................  6
   4.1.  GDR Candidates ..........................................  7
   4.2.  Hash Mask and Hash Algorithm .........................  7
   4.3.  Modulo Hash Algorithm .................................  8
   4.4.  PIM Hello Options .....................................  9
5.  Hello Option Formats .......................................  10
   5.1.  PIM DR Load Balancing Capability (DRLBC) Hello Option ..  10
   5.2.  PIM DR Load Balancing GDR (DRLBGDR) Hello Option .....  10
   6.1.  PIM DR Operation ......................................  12
   6.2.  PIM GDR Candidate Operation ..........................  12
      6.2.1.  Router receives new DRLBGDR ....................  13
      6.2.2.  Router receives updated DRLBGDR ...............  13
   6.3.  PIM Assert Modification ...............................  14
7.  Compatibility ...............................................  15
8.  Manageability Considerations ..............................  15
9.  IANA Considerations ........................................  16
10. Security Considerations ....................................  16
11. Acknowledgement ............................................  16
12. References ..................................................  16
   12.1.  Normative References ................................  16
   12.2.  Informative References ...............................  17
Authors’ Addresses ............................................  17

1.  Introduction

On a multi-access LAN such as an Ethernet, one of the PIM routers is elected as a DR. The PIM DR has two roles in the PIM-SM protocol. On the first hop network, the PIM DR is responsible for registering an active source with the Rendezvous Point (RP) if the group is operating in PIM-SM. On the last hop LAN, the PIM DR is responsible for tracking local multicast listeners and forwarding to these listeners if the group is operating in PIM-SM.
Consider the following last hop LAN in Figure 1:

```
( core networks )
|     |     |
R1   R2   R3
|     |     |
--(last hop LAN)--
|     |
( many receivers )
```

Figure 1: Last Hop LAN

Assume R1 is elected as the Designated Router. According to [RFC4601], R1 will be responsible for forwarding traffic to that LAN on behalf of any local members. In addition to keeping track of IGMP and MLD membership reports, R1 is also responsible for initiating the creation of source and/or shared trees towards the senders or the RPs.

Forcing sole data plane forwarding responsibility on the PIM DR proves a limitation in the protocol. In comparison, even though an OSPF DR, or an IS-IS DIS, handles additional duties while running the OSPF or IS-IS protocols, they are not required to be solely responsible for forwarding packets for the network. On the other hand, on a last hop LAN, only the PIM DR is asked to forward packets while the other routers handle only control traffic (and perhaps drop packets due to RPF failures). The forwarding load of a last hop LAN is concentrated on a single router.

This leads to several issues. One of the issues is that the aggregated bandwidth will be limited to what R1 can handle towards this particular interface. These days, it is very common that the last hop LAN usually consists of switches that run IGMP/MLD or PIM snooping. This allows the forwarding of multicast packets to be restricted only to segments leading to receivers who have indicated their interest in multicast groups using either IGMP or MLD. The emergence of the switched Ethernet allows the aggregated bandwidth to exceed, some times by a large number, that of a single link. For example, let us modify Figure 1 and introduce an Ethernet switch in Figure 2.
Let us assume that each individual link is a Gigabit Ethernet. Each router, R1, R2, and R3, and the switch have enough forwarding capacity to handle hundreds of Gigabits of data.

Let us further assume that each of the hosts requests 500 Mbps of data and different traffic is requested by each host. This represents a total 1.5 Gbps of data, which is under what each switch or the combined uplink bandwidth across the routers can handle, even under failure of a single router.

On the other hand, the link between R1 and switch, via port gi0, can only handle a throughput of 1 Gbps. And if R1 is the only router, the PIM DR elected using the procedure defined by [RFC4601], at least 500 Mbps worth of data will be lost because the only link that can be used to draw the traffic from the routers to the switch is via gi0. In other words, the entire network’s throughput is limited by the single connection between the PIM DR and the switch (or the last hop LAN as in Figure 1).

The problem may also manifest itself in a different way. For example, R1 happens to forward 500 Mbps worth of unicast data to H1, and at the same time, H2 and H3 each requests 300 Mbps of different multicast data. Once again packet drop happens on R1 while in the mean time, there is sufficient forwarding capacity left on R2 and R3 and link capacity between the switch and R2/R3.

Another important issue is related to failover. If R1 is the only forwarder on the last hop router for shared LAN, in the event of a failure when R1 goes out of service, multicast forwarding for the...
In this document, we propose a modification to the PIM-SM protocol that allows more than one of these routers, called Group Designated Router (GDR) to be selected so that the forwarding load can be distributed among a number of routers.

2. Terminology

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

With respect to PIM, this document follows the terminology that has been defined in [RFC4601].

This document also introduces the following new acronyms:

- **GDR**: GDR stands for "Group Designated Router". For each multicast flow, either a (*,G) for ASM, or an (S,G) for SSM, a hash algorithm (described below) is used to select one of the routers as a GDR. The GDR is responsible for initiating the forwarding tree building for the corresponding multicast flow.

- **GDR Candidate**: a last hop router that has potential to become a GDR. A GDR Candidate must have the same DR priority and must run the same GDR election hash algorithm as the DR router. It must send and process new PIM Hello Options as defined in this document. There might be more than one GDR Candidate on a LAN. But only one can become GDR for a specific multicast flow.

3. Applicability

The proposed change described in this specification applies to PIM-SM last hop routers only.

It does not alter the behavior of a PIM DR on the first hop network. This is because the source tree is built using the IP address of the sender, not the IP address of the PIM DR that sends the registers towards the RP. The load balancing between first hop routers can be achieved naturally if an IGP provides equal cost multiple paths (which it usually does in practice). And distributing the load to do registering does not justify the additional complexity required to support it.
4. Functional Overview

In the existing PIM DR election, when multiple last hop routers are connected to a multi-access LAN (for example, an Ethernet), one of them is selected to act as PIM DR. The PIM DR is responsible for sending local Join/Prune messages towards the RP or source. To elect the PIM DR, each PIM router on the LAN examines the received PIM Hello messages and compares its DR priority and IP address with those of its neighbors. The router with the highest DR priority is the PIM DR. If there are multiple such routers, their IP addresses are used as the tie-breaker, as described in [RFC4601].

In order to share forwarding load among last hop routers, besides the normal PIM DR election, the GDR is also elected on the last hop multi-access LAN. There is only one PIM DR on the multi-access LAN, but there might be multiple GDR Candidates.

For each multicast flow, that is (*,G) for ASM and (S,G) for SSM, a hash algorithm is used to select one of the routers to be the GDR. A new DR Load Balancing Capability (DRLBC) PIM Hello Option, which contains hash algorithm type, is announced by routers on interfaces where this specification is enabled. Last hop routers with the new DRLBC Option advertised in its Hello, and using the same GDR election hash algorithm and the same DR priority as the PIM DR, are considered as GDR Candidates.

Hash Masks are defined for Source, Group and RP separately, in order to handle PIM ASM/SSM. The masks, as well as a sorted list of GDR Candidates’ Addresses are announced by DR in a new DR Load Balancing GDR (DRLBGDR) PIM Hello Option.

For each multicast flow, a hash algorithm is used to select one of the routers to be the GDR. The masks are announced in PIM Hello by DR as a DR Load Balancing GDR (DRLBGDR) Hello Option. Besides that, a DR Load Balancing Capability (DRLBC) Hello Option, which contains hash algorithm type, is also announced by the router on interfaces where this specification is enabled. Last hop routers with the new DRLBC Option advertised in its Hello, and using the same GDR election hash algorithm and the same DR priority as the PIM DR, are considered as GDR Candidates.

A hash algorithm based on the announced Source, Group or RP masks allows one GDR to be assigned to a corresponding multicast state. And that GDR is responsible for initiating the creation of the multicast forwarding tree for multicast traffic.
4.1. GDR Candidates

GDR is the new concept introduced by this specification. GDR Candidates are routers eligible for GDR election on the LAN. To become a GDR Candidate, a router MUST support this specification, have the same DR priority and run the same GDR election hash algorithm as the DR on the LAN.

For example, assume there are 4 routers on the LAN: R1, R2, R3 and R4, which all support this specification on the LAN. R1, R2 and R3 have the same DR priority while R4’s DR priority is less preferred. In this example, R4 will not be eligible for GDR election, because R4 will not become a PIM DR unless all of R1, R2 and R3 go out of service.

Further assume router R1 wins the PIM DR election, and R1, R2 run the same hash algorithm for GDR election, while R3 runs a different one. Then only R1 and R2 will be eligible for GDR election, R3 will not.

As a DR, R1 will include its own Load Balancing Hash Masks, and also the identity of R1 and R2 (the GDR Candidates) in its DRLBGDR Hello Option.

4.2. Hash Mask and Hash Algorithm

A Hash Mask is used to extract a number of bits from the corresponding IP address field (32 for v4, 128 for v6), and calculate a hash value. A hash value is used to select a GDR from GDR Candidates advertised by PIM DR. For example, 0.0.255.0 defines a Hash Mask for an IPv4 address that masks the first, the second and the fourth octets.

There are three Hash Masks defined,

- RP Hash Mask
- Source Hash Mask
- Group Hash Mask

The hash masks need to be configured on the PIM routers that can potentially become a PIM DR, unless the implementation provides default hash mask. An implementation SHOULD provide masks with default values 255.255.255.255 (IPv4) and FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF (IPv6).
If the group is ASM, and if the RP Hash Mask announced by the PIM DR is not 0, calculate the value of hashvalue_RP [Section 4.3] to determine GDR.

If the group is ASM and if the RP Hash Mask announced by the PIM DR is 0, obtain the value of hashvalue_Group [Section 4.3] to determine GDR.

If the group is SSM, use hashvalue_SG [Section 4.3] to determine GDR.

A simple Modulo hash algorithm will be discussed in this document. However, to allow other hash algorithm to be used, a 4-bytes "Hash Algorithm Type" field is included in DRLBGDR Hello Option to specify the hash algorithm used by a last hop router.

If different hash algorithm types are advertised among last hop routers, only last hop routers running the same hash algorithm as the DR (and having the same DR priority as the DR) are eligible for GDR election.

4.3. Modulo Hash Algorithm

Modulo hash algorithm is discussed here as an example, with detailed description on hashvalue_RP.

- For ASM groups, with a non-zero RP_hash mask, hash value is calculated as:

  \[\text{hashvalue}_R = ((\text{RP}_\text{address} \& \text{RP}_\text{hashmask}) >> N) \& 0xFFFF \mod M\]

  RP_address is the address of the RP defined for the group. N is the number of zeros, counted from the least significant bit of the RP_hashmask. M is the number of GDR Candidates.

  For example, Router X with IPv4 address 203.0.113.1, receives a DRLBGDR Hello Option from the DR, which announces RP Hash Mask 0.0.255.0, and a list of GDR Candidates, sorted by IP addresses from high to low, 203.0.113.3, 203.0.113.2 and 203.0.113.1. The ordinal number assigned to those addresses would be:

  0 for 203.0.113.3; 1 for 203.0.113.2; 2 for 203.0.113.1 (Router X)

  Assume there are 2 RPs: RP1 192.0.2.1 for Group1 and RP2 198.51.100.2 for Group2. Following the modulo hash algorithm:
N is 8 for 0.0.255.0, and M is 3 for the total number of GDR Candidates. The hashvalue_RP for RP1 192.0.2.1 is:

```
(((192.0.2.1 & 0.0.255.0) >> 8) & 0xFFFF % 3) = 2 % 3 = 2
```

matches the ordinal number assigned to Router X. Router X will be the GDR for Group1, which uses 192.0.2.1 as the RP.

The hashvalue_RP for RP2 198.51.100.2 is:

```
(((198.51.100.2 & 0.0.255.0) >> 8) & 0xFFFF % 3) = 100 % 3 = 1
```

which is different from Router X's ordinal number 2, hence, Router X will not be GDR for Group2.

- If RP_hashmask is 0, a hash value for ASM group is calculated using the group Hash Mask:

```
hashvalue_Group = (((Group_address & Group_hashmask) >> N) & 0xFFFF) % M
```

Compare hashvalue_Group with Ordinal number assigned to Router X, to decide if Router X is the GDR.

- For SSM groups, a hash value is calculated using both the source and group Hash Mask

```
hashvalue_SG = ((((Source_address & Source_hashmask) >> N_S) & 0xFFFF) ^ (((Group_address & Group_hashmask) >> N_G) & 0xFFFF)) % M
```

### 4.4. PIM Hello Options

When a last hop PIM router sends a PIM Hello from an interface with this specification enabled, it includes a new option, called "Load Balancing Capability (DRLBC)".

Besides this DRLBC Hello Option, the elected PIM DR also includes a new "DR Load Balancing GDR (DRLBGDR) Hello Option". The DRLBGDR Hello Option consists of three Hash Masks as defined above and also the sorted list of all GDR Candidates’ Address on the last hop LAN.

The elected PIM DR uses DRLBC Hello Option advertised by all routers on the last hop LAN to compose its DRLBGDR. The GDR Candidates use DRLBGDR Hello Option advertised by PIM DR to calculate hash value.
5. Hello Option Formats

5.1. PIM DR Load Balancing Capability (DRLBC) Hello Option

Figure 3: Capability Hello Option

Type: TBD
Length: 4 octets
Hash Algorithm Type: 0 for Modulo hash algorithm

This DRLBC Hello Option SHOULD be advertised by last hop routers from interfaces with this specification enabled.

5.2. PIM DR Load Balancing GDR (DRLBGDR) Hello Option

Figure 4: GDR Hello Option

Type: TBD
Length: 3 x (4 byte or 16 byte) + n x (4 byte or 16 byte) where n is number of GDR candidate
Group Mask (32/128 bits): Mask
Source Mask (32/128 bits): Mask
RP Mask (32/128 bits): Mask

All masks MUST be in the same address family as the Hello IP header.

GDR Address (32/128 bits): Address(es) of GDR Candidate(s)

All addresses must be in the same address family as the Hello IP header. The addresses are sorted from high to low. The order is converted to the ordinal number associated with each GDR candidate in hash value calculation. For example, addresses advertised are R3, R2, R1, the ordinal number assigned to R3 is 0, to R2 is 1 and to R1 is 2.

If "Interface ID" option, as described in [RFC6395], presents in a GDR Candidate’s PIM Hello message, and the "Router ID" portion is non-zero,

+ For IPv4, the "GDR Candidate Address" will be set directly to "Router ID".

+ For IPv6, the "GDR Candidate Address" will be set to the IPv4-IPv6 translated address of "Router ID", as described in [RFC4291], that is the "Router-ID" is appended to the prefix of 96-bits zeros.

If the "Interface ID" option is not present in a GDR Candidate’s PIM Hello message, or if the "Interface ID" option is present, but "Router ID" field is zero, the "GDR Candidate Address" will be the IPv4 or IPv6 source address from PIM Hello message.

This DRLBGDR Hello Option SHOULD only be advertised by the elected PIM DR.

6.1.  PIM DR Operation

The DR election process is still the same as defined in [RFC4601]. A DR that has this specification enabled on the interface, advertises the new LBGRD Hello Option, which contains value of masks from user configuration, followed by a sorted list of all GDR Candidates’ Addresses, from high to low. Moreover, same as non-DR routers, DR also advertises DRLBC Hello Option to indicate its capability of supporting this specification and the type of its GDR election hash algorithm.

If a PIM DR receives a PIM Hello with DRLBGRD Option, the PIM DR SHOULD ignore the TLV.

If a PIM DR receives a neighbor DRLBC Hello Option, which contains the same hash algorithm type as the DR, and the neighbor has the same DR priority as the DR, PIM DR SHOULD consider the neighbor as a GDR Candidate and insert the GDR Candidate’s Address into the sorted list of DRLBGRD Option.

6.2.  PIM GDR Candidate Operation

When an IGMP/MLD join is received, without this proposal, only PIM DR will handle the join and potentially run into the issues described earlier. Using this proposal, a hash algorithm is used on GDR Candidate to determine which router is going to be responsible for building forwarding trees on behalf of the host.

A router which supports this specification, a interface where this protocol is enabled advertises DRLBC Hello Option in its PIM Hello, even if the router may not be considered as a GDR Candidate, for example, due to low DR priority. once DR election is done, DRLBGDR Hello option would be received from the current PIM DR on link.

A GDR Candidate may receive a DRLBGDR Hello Option from PIM DR, with different Hash Masks from those configured on it, The GDR Candidate must use the Hash Masks advertised by the PIM DR to calculate the hash value.

A GDR Candidate may receive a DRLBGDR Hello Option from a PIM router which is not DR. The GDR Candidate MUST ignore such DRLBGDR Hello Option.

A GDR Candidate may receive a Hello from the elected PIM DR, and the PIM DR does not support this specification. The GDR election described by this specification will not take place, that is only the PIM DR joins the multicast tree.
A router only act as GDR if it is included in the GDR list of DRLBGDR Hello Option

6.2.1. Router receives new DRLBGDR

When a router receives new DRLBGDR from the current PIM DR, it need to process and check if router is in list of GDR

1. If router is not listed as a GDR candidate in DRLBGDR, no action needed.

2. If router is listed as a GDR candidate in DRLBGDR, then it need to process each of the groups in the IGMP/MLD reports. The masks are announced in the PIM Hello by DR as DRLBGDR Hello option. For each of groups in the reports it need to run hash algorithm (described in section 4.3) based on the announced Source, Group or RP masks to determine if it is GDR for specified group. If hash result is to be GDR for multicast flow, it does build multicast forwarding tree. If it is not GDR for flow, no action is needed.

6.2.2. Router receives updated DRLBGDR

If router (GDR or non GDR) receives an unchanged DRLBGDR from the current PIM DR, no action needed.

If router (GDR or non GDR) receives a new or modified DRLBGDR from the current PIM DR. It requires processing as described below

1. If it was GDR and still included in current GDR list: It need to process each of the groups, run hash algorithm to check if it is still GDR for given group.

   If it was GDR for group earlier, and even new hash choose it as GDR, no processing required.

   If it was GDR for group earlier and now it is no more GDR, then it sets its assert metric for this flow to be (PIM_ASSERT_INFINITY - 1), as explained in Sec 6.3

   If it was not GDR for group earlier, and even new hash does not make it GDR no processing required.

   If it was not GDR earlier and now becomes GDR, it starts building multicast forwarding tree for this flow.

2. If it was non GDR, and updated DRLBGDR from current PIM DR contains this router as one of the GDR. In this case this router
being new GDR candidate MUST run hash algorithm for each of the groups (multicast flows) and for given group,

If it is not GDR, no processing is required.

If it is hashed as GDR, it needs to build multicast forwarding tree.

3. If a router receives IGMP/MLD report for which the router has been the GDR AND the DRLBGDR has changed since last report for this flow, then the router MUST determine if it is still the GDR. if it is, no action needed. if it is not, then the router sets its assert metric for this flow to be (PIM_ASSERT_INFINITY - 1) as explained in Sec 6.3.

6.3. PIM Assert Modification

It is possible that the identity of the GDR might change in the middle of an active flow. Examples this could happen include:

When a new PIM router comes up

When a GDR restarts

When the GDR changes, existing traffic might be disrupted. Duplicates or packet loss might be observed. To illustrate the case, consider the following scenario: there are two streams G1 and G2. R1 is the GDR for G1, and R2 is the GDR for G2. When R3 comes up online, it is possible that R3 becomes GDR for both G1 and G2, hence R3 starts to build the forwarding tree for G1 and G2. If R1 and R2 stop forwarding before R3 completes the process, packet loss might occur. On the other hand, if R1 and R2 continue forwarding while R3 is building the forwarding tree, duplicates might occur.

This is not a typical deployment scenario but it still might happen. Here we describe a mechanism to minimize the impact. The motivation is that we want to minimize packet loss. And therefore, we would allow a small amount of duplicates and depend on PIM Assert to minimize the duplication.

When the role of GDR changes as above, instead of immediately stopping forwarding, R1 and R2 continue forwarding to G1 and G2 respectively, while at the same time, R3 build forwarding trees for G1 and G2. This will lead to PIM Asserts.

With introduction of GDR, the following modification to the Assert packet MUST be done: if a router enables this specification on its downstream interface, but it is not a GDR (before network event it
was GDR), it would adjust its Assert metric to (PIM_ASSERT_INFINITY - 1).

Using the above example, for G1, assume R1 and R3 agree on the new GDR, which is R3. R1 will set its Assert metric as (PIM_ASSERT_INFINITY - 1). That will make R3, which has normal metric in its Assert as the Assert winner.

For G2, assume it takes a little bit longer time for R2 to find out that R3 is the new GDR and still thinks itself being the GDR while R3 already has assumed the role of GDR. Since both R2 and R3 think they are GDRs, they further compare the metric and IP address. If R3 has the better routing metric, or same metric but better tie-breaker, the result will be consistent with GDR selection. If unfortunately, R2 has the better metric or same metric but better tie-breaker R2 will become the Assert winner and continues to forward traffic. This will continue until:

The next PIM Hello option from DR is seen that selects R3 as the GDR. R3 will then build the forwarding tree and send an Assert.

The process continues until R2 agrees to the selection of R3 as being the GDR, and set its own Assert metric to (PIM_ASSERT_INFINITY - 1), which will make R3 the Assert winner. During the process, we will see intermittent duplication of traffic but packet loss will be minimized. In the unlikely case that R2 never relinquishes its role as GDR (while every other router thinks otherwise), the proposed mechanism also helps to keep the duplication to a minimum until manual intervention takes place to remedy the situation.

7. Compatibility

In case of hybrid Ethernet shared LAN (where some PIM router enables specification defined in this draft and some do not enable)

  o If router which does not support specification defined in this draft becomes DR on link, it MUST be only DR on link as [RFC4601] and there would be no router which would act as GDR.

  o If router which does not support specification defined in this draft becomes non DR on link, then it should act as non-DR defined in [RFC4601].

8. Manageability Considerations

  o All of the router in LAN who are supporting this specification MUST use identical Hash Algorithm Type (described in section 5.1). In case of hybrid Hash Algorithm Type router must go backward to
use DR election method defined in PIM-SM [RFC4601]. Migration between different algorithm type is out of scope of this document.

9. IANA Considerations

Two new PIM Hello Option Types have been assigned to the DR Load Balancing messages. [HELLO-OPT], this document recommends 34(0x22) as the new "PIM DR Load Balancing Capability Hello Option", and 35(0x23) as the new "PIM DR Load Balancing GDR Hello Option".

10. Security Considerations

Security of the new DR Load Balancing PIM Hello Options is only guaranteed by the security of PIM Hello message, so the security considerations for PIM Hello messages as described in PIM-SM [RFC4601] apply here.

11. Acknowledgement

The authors would like to thank Steve Simlo, Taki Millonis for helping with the original idea, Bill Atwood, Bharat Joshi for review comments, Toerless Eckert and Rishabh Parekh for helpful conversation on the document.

12. References

12.1. Normative References


12.2. Informative References

[HELLO-OPT]

Authors' Addresses

Yiqun Cai
Alibaba Group

Email: yiqun.cai@alibaba-inc.com

Heidi Ou
Alibaba Group

Sri Vallepalli
Cisco Systems
3625 Cisco Way,
Sanjose, CALIFORNIA 95134
UNITED STATES

Email: svallepa@cisco.com

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

Email: mankamis@cisco.com

Stig Venaas
Cisco Systems
821 Alder Drive,
MILPITAS, CALIFORNIA 95035
UNITED STATES

Email: stig@cisco.com
Abstract

On a multi-access network, one of the PIM routers is elected as a Designated Router (DR). On the last hop LAN, the PIM DR is responsible for tracking local multicast listeners and forwarding traffic to these listeners if the group is operating in PIM-SM. In this document, we propose a modification to the PIM-SM protocol that allows more than one of these last hop routers to be selected so that the forwarding load can be distributed among these routers.

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 December 21, 2018.

Copyright Notice

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

On a multi-access LAN such as an Ethernet, one of the PIM routers is elected as a DR. The PIM DR has two roles in the PIM-SM protocol. On the first hop network, the PIM DR is responsible for registering an active source with the Rendezvous Point (RP) if the group is operating in PIM-SM. On the last hop LAN, the PIM DR is responsible for tracking local multicast listeners and forwarding to these listeners if the group is operating in PIM-SM.
Consider the following last hop LAN in Figure 1:

```
( core networks )
|     |     |
|     |     |
R1    R2     R3
|     |     |
--(last hop LAN)--
|     |     |
( many receivers )
```

Figure 1: Last Hop LAN

Assume R1 is elected as the Designated Router. According to [RFC4601], R1 will be responsible for forwarding traffic to that LAN on behalf of any local members. In addition to keeping track of IGMP and MLD membership reports, R1 is also responsible for initiating the creation of source and/or shared trees towards the senders or the RPs.

Forcing sole data plane forwarding responsibility on the PIM DR uncovers a limitation in the protocol. In comparison, even though an OSPF DR or an IS-IS DIS handles additional duties while running the OSPF or IS-IS protocols, they are not required to be solely responsible for forwarding packets for the network. On the other hand, on a last hop LAN, only the PIM DR is asked to forward packets while the other routers handle only control traffic (and perhaps drop packets due to RPF failures). Hence the forwarding load of a last hop LAN is concentrated on a single router.

This leads to several issues. One of the issues is that the aggregated bandwidth will be limited to what R1 can handle towards this particular interface. It is very common that the last hop LAN usually consists of switches that run IGMP/MLD or PIM snooping. This allows the forwarding of multicast packets to be restricted only to segments leading to receivers who have indicated their interest in multicast groups using either IGMP or MLD. The emergence of the switched Ethernet allows the aggregated bandwidth to exceed, sometimes by a large number, that of a single link. For example, let us modify Figure 1 and introduce an Ethernet switch in Figure 2.
Let us assume that each individual link is a Gigabit Ethernet. Each router, R1, R2 and R3, and the switch have enough forwarding capacity to handle hundreds of Gigabits of data.

Let us further assume that each of the hosts requests 500 Mbps of unique multicast data. This totals to 1.5 Gbps of data, which is less than what each switch or the combined uplink bandwidth across the routers can handle, even under failure of a single router.

On the other hand, the link between R1 and switch, via port gi0, can only handle a throughput of 1Gbps. And if R1 is the only DR (the PIM DR elected using the procedure defined by [RFC4601]) at least 500 Mbps worth of data will be lost because the only link that can be used to draw the traffic from the routers to the switch is via gi0. In other words, the entire network’s throughput is limited by the single connection between the PIM DR and the switch (or the last hop LAN as in Figure 1).

The problem may also manifest itself in a different way. For example, R1 happens to forward 500 Mbps worth of unicast data to H1, and at the same time, H2 and H3 each request 300 Mbps of different multicast data. R1 experiences packet drop once again. While, in the meantime, there is sufficient forwarding capacity left on R2 and R3 and unused link capacity between the switch and R2/R3.

Another important issue is related to failover. If R1 is the only forwarder on the last hop router for shared LAN, when R1 goes out of service, multicast forwarding for the entire LAN has to be rebuilt by the newly elected PIM DR. However, if there was a way that allowed...
multiple routers to forward to the LAN for different groups, failure of one of the routers would only lead to disruption to a subset of the flows, therefore improving the overall resilience of the network.

There is limitation in the hash algorithm used in this document, but this draft provides the option to have different and more consistent hash algorithms in the future.

In this document, we propose a modification to the PIM-SM protocol that allows more than one of these routers, called Group Designated Routers (GDR) to be selected so that the forwarding load can be distributed among a number of routers.

2. Terminology

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

With respect to PIM, this document follows the terminology that has been defined in [RFC4601].

This document also introduces the following new acronyms:

- **GDR**: GDR stands for "Group Designated Router". For each multicast flow, either a (*,G) for ASM, or an (S,G) for SSM, a hash algorithm (described below) is used to select one of the routers as a GDR. The GDR is responsible for initiating the forwarding tree building process for the corresponding multicast flow.

- **GDR Candidate**: a last hop router that has the potential to become a GDR. A GDR Candidate must have the same DR priority and must run the same GDR election hash algorithm as the DR router. It must send and process new PIM Hello Options as defined in this document. There might be more than one GDR Candidate on a LAN, but only one can become GDR for a specific multicast flow.

3. Applicability

The proposed change described in this specification applies to PIM-SM last hop routers only.

It does not alter the behavior of a PIM DR on the first hop network. This is because the source tree is built using the IP address of the sender, not the IP address of the PIM DR that sends the registers towards the RP. The load balancing between first hop routers can be achieved naturally if an IGP provides equal cost multiple paths (which it usually does in practice). Also distributing the load to
do registering does not justify the additional complexity required to support it.

4.  Functional Overview

In the existing PIM DR election, when multiple last hop routers are connected to a multi-access LAN (for example, an Ethernet), one of them is selected to act as PIM DR. The PIM DR is responsible for sending local Join/Prune messages towards the RP or source. In order to elect the PIM DR, each PIM router on the LAN examines the received PIM Hello messages and compares its DR priority and IP address with those of its neighbors. The router with the highest DR priority is the PIM DR. If there are multiple such routers, their IP addresses are used as the tie-breaker, as described in [RFC4601].

In order to share forwarding load among last hop routers, besides the normal PIM DR election, the GDR is also elected on the last hop multi-access LAN. There is only one PIM DR on the multi-access LAN, but there might be multiple GDR Candidates.

For each multicast flow, that is, (*,G) for ASM and (S,G) for SSM, a hash algorithm is used to select one of the routers to be the GDR. A new DR Load Balancing Capability (DRLBC) PIM Hello Option, which contains hash algorithm type, is announced by routers on interfaces where this specification is enabled. Last hop routers with the new DRLBC Option advertised in its Hello, and using the same GDR election hash algorithm and the same DR priority as the PIM DR, are considered as GDR Candidates.

Hash Masks are defined for Source, Group and RP separately, in order to handle PIM ASM/SSM. The masks, as well as a sorted list of GDR Candidates’ Addresses, are announced by DR in a new DR Load Balancing GDR (DRLBGDR) PIM Hello Option.

A hash algorithm based on the announced Source, Group, or RP masks allows one GDR to be assigned to a corresponding multicast state. And that GDR is responsible for initiating the creation of the multicast forwarding tree for multicast traffic.

4.1.  GDR Candidates

GDR is the new concept introduced by this specification. GDR Candidates are routers eligible for GDR election on the LAN. To become a GDR Candidate, a router MUST support this specification, have the same DR priority and run the same GDR election hash algorithm as the DR on the LAN.
For example, assume there are 4 routers on the LAN: R1, R2, R3 and R4, which all support this specification. R1, R2 and R3 have the same DR priority while R4’s DR priority is less preferred. In this example, R4 will not be eligible for GDR election, because R4 will not become a PIM DR unless all of R1, R2 and R3 go out of service.

Furthermore, assume router R1 wins the PIM DR election, R1 and R2 run the same hash algorithm for GDR election, while R3 runs a different one. In this case, only R1 and R2 will be eligible for GDR election, while R3 will not.

As a DR, R1 will include its own Load Balancing Hash Masks and the identity of R1 and R2 (the GDR Candidates) in its DRLBGDR Hello Option.

4.2. Hash Mask and Hash Algorithm

A Hash Mask is used to extract a number of bits from the corresponding IP address field (32 for v4, 128 for v6) and calculate a hash value. A hash value is used to select a GDR from GDR Candidates advertised by PIM DR. For example, 0.0.255.0 defines a Hash Mask for an IPv4 address that masks the first, the second, and the fourth octets.

There are three Hash Masks defined,

- RP Hash Mask
- Source Hash Mask
- Group Hash Mask

The hash masks need to be configured on the PIM routers that can potentially become a PIM DR, unless the implementation provides default Hash Mask. An implementation SHOULD provide masks with default values 255.255.255.255 (IPv4) and FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF (IPv6).

- If the group is ASM and the RP Hash Mask announced by the PIM DR is not 0, calculate the value of hashvalue_RP [Section 4.3] to determine GDR.
- If the group is ASM and the RP Hash Mask announced by the PIM DR is 0, obtain the value of hashvalue_Group [Section 4.3] to determine GDR.
- If the group is SSM, use hashvalue_SG [Section 4.3] to determine GDR.
A simple Modulo hash algorithm will be discussed in this document. However, to allow another hash algorithms to be used, a 4-bytes "Hash Algorithm Type" field is included in DRLBC Hello Option to specify the hash algorithm used by a last hop router.

If different hash algorithm types are advertised among last hop routers, only last hop routers running the same hash algorithm as the DR (and having the same DR priority as the DR) are eligible for GDR election.

4.3. Modulo Hash Algorithm

Modulo hash algorithm is discussed here with a detailed description on hashvalue_RP. The same algorithm is described in brief for hashvalue_Group using the group address instead of the RP address for an ASM group with RP_hashmask==0, and also with hashvalue_SG for a the source address of an (S,G), instead of the RP address,

- For ASM groups, with a non-zero RP_Hash Mask, hash value is calculated as:

  \[
  \text{hashvalue}_\text{RP} = (((\text{RP}_\text{address} \& \text{RP}_\text{hashmask}) >> N) \& \text{0xFFFF}) \% M
  \]

  RP_address is the address of the RP defined for the group. N is the number of zeros, counted from the least significant bit of the RP_hashmask. M is the number of GDR Candidates.

  For example, Router X with IPv4 address 203.0.113.1 receives a DRLBGDR Hello Option from the DR, which announces RP Hash Mask 0.0.255.0 and a list of GDR Candidates, sorted by IP addresses from high to low: 203.0.113.3, 203.0.113.2 and 203.0.113.1. The ordinal number assigned to those addresses would be:

  0 for 203.0.113.3; 1 for 203.0.113.2; 2 for 203.0.113.1 (Router X)

  Assume there are 2 RPs: RP1 192.0.2.1 for Group1 and RP2 198.51.100.2 for Group2. Following the modulo hash algorithm:

  N is 8 for 0.0.255.0, and M is 3 for the total number of GDR Candidates. The hashvalue_RP for RP1 192.0.2.1 is:

  \[
  (((192.0.2.1 \& 0.0.255.0) >> 8) \& \text{0xFFFF} \% 3) = 2 \% 3 = 2
  \]

  matches the ordinal number assigned to Router X. Router X will be the GDR for Group1, which uses 192.0.2.1 as the RP.

  The hashvalue_RP for RP2 198.51.100.2 is:
which is different from Router X’s ordinal number(2) hence, Router X will not be GDR for Group2.

- If RP_hashmask is 0, a hash value for ASM group is calculated using the group Hash Mask:

\[
\text{hashvalue}_\text{Group} = (((\text{Group_address} \& \text{Group_hashmask}) \gg N) \& 0xFFFF) \mod M
\]

Compare hashvalue_Group with Ordinal number assigned to Router X, to decide if Router X is the GDR.

- For SSM groups, a hash value is calculated using both the source and group Hash Mask:

\[
\text{hashvalue}_\text{SG} = (((\text{Source_address} \& \text{Source_hashmask}) \gg N_S) \& 0xFFFF) \oplus (((\text{Group_address} \& \text{Group_hashmask}) \gg N_G) \& 0xFFFF) \mod M
\]

### 4.4. PIM Hello Options

When a last hop PIM router sends a PIM Hello from an interface with this specification enabled, it includes a new option, called "Load Balancing Capability (DRLBC)".

Besides this DRLBC Hello Option, the elected PIM DR also includes a new "DR Load Balancing GDR (DRLBGDR) Hello Option". The DRLBGDR Hello Option consists of three Hash Masks as defined above and also the sorted list of all GDR Candidates’ Address on the last hop LAN.

The elected PIM DR uses DRLBC Hello Option advertised by all routers on the last hop LAN to compose its DRLBGDR. The GDR Candidates use DRLBGDR Hello Option advertised by PIM DR to calculate hash value.

### 5. Hello Option Formats

#### 5.1. PIM DR Load Balancing Capability (DRLBC) Hello Option
Figure 3: Capability Hello Option

Type: TBD.
Length: 4 octets
Hash Algorithm Type: 0 for Modulo hash algorithm

This DRLBC Hello Option SHOULD be advertised by last hop routers from interfaces with this specification enabled.

5.2. PIM DR Load Balancing GDR (DRLBGDR) Hello Option

Type: TBD

Length: 3 x (4 byte or 16 byte) + n x (4 byte or 16 byte) where n is the number of GDR candidates.

Group Mask (32/128 bits): Mask

Source Mask (32/128 bits): Mask
RP Mask (32/128 bits): Mask

All masks MUST be in the same address family as the Hello IP header.

GDR Address (32/128 bits): Address(es) of GDR Candidate(s)

All addresses must be in the same address family as the Hello IP header. The addresses are sorted in descending order. The order is converted to the ordinal number associated with each GDR candidate in hash value calculation. For example, addresses advertised are R3, R2, R1, the ordinal number assigned to R3 is 0, to R2 is 1 and to R1 is 2.

If "Interface ID" option, as described in [RFC6395], presents in a GDR Candidate’s PIM Hello message, and the "Router ID" portion is non-zero,

+ For IPv4, the "GDR Candidate Address" will be set directly to "Router ID".

+ For IPv6, the "GDR Candidate Address" will be set to the IPv4-IPv6 translated address of "Router ID", as described in [RFC4291], that is the "Router-ID" is appended to the prefix of 96-bits zeros.

If the "Interface ID" option is not present in a GDR Candidate’s PIM Hello message, or if the "Interface ID" option is present but the "Router ID" field is zero, the "GDR Candidate Address" will be the IPv4 or IPv6 source address from PIM Hello message.

This DRLBGDR Hello Option MUST only be advertised by the elected PIM DR.


6.1. PIM DR Operation

The DR election process is still the same as defined in [RFC4601]. A DR that has this specification enabled on the interface advertises the new DRLBGDR Hello Option, which contains value of masks from user configuration, followed by a sorted list of all GDR Candidates’ Addresses, from the highest value to the lowest value. Moreover,
same as non-DR routers, DR also advertises DRLBC Hello Option to indicate its capability of supporting this specification and the type of its GDR election hash algorithm.

If a PIM DR receives a PIM Hello with DRLBGDR Option, the PIM DR SHOULD ignore the TLV.

If a PIM DR receives a neighbor DRLBC Hello Option, which contains the same hash algorithm type as the DR, and the neighbor has the same DR priority as the DR, PIM DR SHOULD consider the neighbor as a GDR Candidate and insert the GDR Candidate’s Address into the sorted list of DRLBGDR Option.

6.2. PIM GDR Candidate Operation

When an IGMP/MLD join is received, without this specification, only PIM DR will handle the join and potentially run into the issues described earlier. Using this specification, a hash algorithm is used on GDR Candidate to determine which router is going to be responsible for building forwarding trees on behalf of the host.

If a router supports this specification then each of the interfaces where multicast protocol is enabled, it MUST advertise DRLBC Hello Option in its PIM Hello. Though DRLBC option in PIM hello does not guarantee that this router would be considered as a GDR candidate. For example, this router may have lower priority configured on shared LAN compare to other PIM routers. Once DR election is done, DRLBGDR Hello option would be received from the current PIM DR on the link which would contain list of GDR.

A GDR Candidate may receive a DRLBGDR Hello Option from PIM DR with different Hash Masks from those configured on it. The GDR Candidate must use the Hash Masks advertised by the PIM DR to calculate the hash value.

A GDR Candidate may receive a DRLBGDR Hello Option from a PIM router which is not DR. The GDR Candidate MUST ignore such DRLBGDR Hello Option.

A GDR Candidate may receive a Hello from the elected PIM DR, and the PIM DR does not support this specification. The GDR election described by this specification will not take place, that is only the PIM DR joins the multicast tree.

A router only acts as GDR if it is included in the GDR list of DRLBGDR Hello Option.
6.2.1. Router Receives New DRLBGDR

When a router receives a new DRLBGDR from the current PIM DR, it need to process and check if router is in list of of GDR

1. If a router is not listed as a GDR candidate in DRLBGDR, no action is needed.

2. If a router is listed as a GDR candidate in DRLBGDR, then it need to process each of the groups in the IGMP/MLD reports. The masks are announced in the PIM Hello by DR as DRLBGDR Hello option. For each of groups in the reports it (PIM Router) needs to run hash algorithm (described in section 4.3) based on the announced Source, Group or RP masks to determine if it is GDR for specified group. If the hash result is to be the GDR for the multicast flow, it does build the multicast forwarding tree. If it is not the GDR for the multicast flow, no action is needed.

6.2.2. Router Receives Updated DRLBGDR

If a router (GDR or non GDR) receives an unchanged DRLBGDR from the current PIM DR, no action is needed.

If a router (GDR or non GDR) receives a new or modified DRLBGDR from the current PIM DR. It requires processing as described below:

1. If it was GDR and still included in current GDR list: it needs to process each of the groups and run the hash algorithm to check if it is still the GDR for the given group.

   If it was the GDR for group G and the new hash result chose it as the GDR, then no processing is required.

   If it was the GDR for a group earlier and now it is no longer the GDR, then it sets its assert metric for the multicast flow to be (PIM_ASSERT_INFINITY - 1), as explained in Sec 6.3

   If it was not the GDR for a group earlier, than even the new hash does not make it GDR. For the multicast group no processing is required.

   If it was not the GDR for an earlier group and now becomes the GDR, it starts building multicast forwarding tree for this flow.

2. If it was not the GDR, and updated DRLBGDR from current PIM DR contains this router as one of the GDR. In this case this router
being new GDR candidate MUST run hash algorithm for each of the groups (multicast flows) and for given group,

If it is not the GDR, no processing is required.

If it is hashed as the GDR, it needs to build multicast forwarding tree.

6.3. PIM Assert Modification

It is possible that the identity of the GDR might change in the middle of an active flow. Examples this could happen include:

When a new PIM router comes up

When a GDR restarts

When the GDR changes, existing traffic might be disrupted. Duplicates or packet losses might be observed. To illustrate the case, consider the following scenario where there are two streams G1 and G2. R1 is the GDR for G1, and R2 is the GDR for G2. When R3 comes up online, it is possible that R3 becomes GDR for both G1 and G2, hence R3 starts to build the forwarding tree for G1 and G2. If R1 and R2 stop forwarding before R3 completes the process, packet loss might occur. On the other hand, if R1 and R2 continue forwarding while R3 is building the forwarding trees, duplicates might occur.

This is not a typical deployment scenario but might still happen. Here we describe a mechanism to minimize the impact. We essentially want to minimize packet loss. Therefore, we would allow a small amount of duplicates and depend on PIM Assert to minimize the duplication.

When the role of GDR changes as above, instead of immediately stopping forwarding, R1 and R2 continue forwarding to G1 and G2 respectively, while, at the same time, R3 build forwarding trees for G1 and G2. This will lead to PIM Asserts.

With the introduction of GDR, the following modification to the Assert packet MUST be done: if a router enables this specification on its downstream interface, but it is not a GDR (before network event it was GDR), it would adjust its Assert metric to (PIM_ASSERT_INFINITY – 1).

Using the above example, for G1, assume R1 and R3 agree on the new GDR, which is R3. R1 will set its Assert metric as
(PIM_ASSUME_INFINITY - 1). That will make R3, which has normal metric in its Assert as the Assert winner.

For G2, assume it takes a slightly longer time for R2 to find out that R3 is the new GDR and still considers itself being the GDR while R3 already has assumed the role of GDR. Since both R2 and R3 think they are GDRs, they further compare the metric and IP address. If R3 has the better routing metric, or the same metric but a better tie-breaker, the result will be consistent during GDR selection. If unfortunately, R2 has the better metric or the same metric but a better tie-breaker, R2 will become the Assert winner and continues to forward traffic. This will continue until:

The next PIM Hello option from DR selects R3 as the GDR. R3 will then build the forwarding tree and send an Assert.

The process continues until R2 agrees to the selection of R3 as the GDR, and set its own Assert metric to (PIM_ASSUME_INFINITY - 1), which will make R3 the Assert winner. During the process, we will see intermittent duplication of traffic but packet loss will be minimized. In the unlikely case that R2 never relinquishes its role as GDR (while every other router thinks otherwise), the proposed mechanism also helps to keep the duplication to a minimum until manual intervention takes place to remedy the situation.

7. Compatibility

In case of the hybrid Ethernet shared LAN (where some PIM router enables specification defined in this draft and some do not enable)

- If a router which does not support specification defined in this draft becomes DR on link, it MUST be only DR on link as [RFC4601] and there would be no router which would act as GDR.

- If a router which does not support specification defined in this draft becomes non DR on link, then it should act as non-DR defined in [RFC4601].

8. Manageability Considerations

- All of the routers in LAN that support this specification MUST use identical Hash Algorithm Type (described in section 5.1). In the case of a hybrid Hash Algorithm Type, one MUST go backward to use DR election method defined in PIM-SM [RFC4601]. Migration between different algorithm type is out of the scope of this document.
9. IANA Considerations

IANA has temporarily assigned type 34 for the PIM DR Load Balancing Capability (DRLBC) Hello Option, and type 35 for the PIM DR Load Balancing GDR (DRLBGDR) Hello Option. IANA is requested to make these assignments permanent when this document is published as an RFC. The string TBD should be replaced by the assigned values accordingly.

10. Security Considerations

Security of the new DR Load Balancing PIM Hello Options is only guaranteed by the security of PIM Hello message, so the security considerations for PIM Hello messages as described in PIM-SM [RFC4601] apply here.

11. Acknowledgement

The authors would like to thank Steve Simlo, Taki Millonis for helping with the original idea, Bill Atwood, Bharat Joshi for review comments, Toerless Eckert and Rishabh Parekh for helpful conversation on the document.

Special thanks to Anish Kachinthaya, Anvitha Kachinthaya and Jake Holland for reviewing the document and providing comments.

12. References

12.1. Normative References


12.2. Informative References

[HELLO-OPT]

Authors' Addresses

Yiqun Cai
Alibaba Group
Email: yiqun.cai@alibaba-inc.com

Heidi Ou
Alibaba Group

Sri Vallepalli
Cisco Systems
3625 Cisco Way,
Sanjose, CALIFORNIA 95134
UNITED STATES
Email: svallepa@cisco.com

Mankamana Mishra
Cisco Systems
821 Alder Drive,
MILPITAS, CALIFORNIA 95035
UNITED STATES
Email: mankamis@cisco.com

Stig Venaas
Cisco Systems
821 Alder Drive,
MILPITAS, CALIFORNIA 95035
UNITED STATES
Email: stig@cisco.com
Andy Green
British Telecom
Adastral Park
Ipswich IP5 2RE
United Kingdom

Email: andy.da.green@bt.com
A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)
draft-ietf-pim-igmp-mld-yang-06

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 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 (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents.
1. Introduction

YANG [RFC6020] [RFC7950] is a data definition language that was introduced to model the configuration and running state of a device managed using NETCONF [RFC6241]. 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. Requirements Language

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

1.2. Terminology

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

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

2. Design of Data model

2.1. Scope of model

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

The 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 follows the Guidelines for YANG Module Authors (NMDA) [draft-dsdt-nmda-guidelines-01]. The IGMP module defines the routing-control-plane-protocol-wide configuration and operational state options separately in a three-level hierarchy as listed below:

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

Interface-global: Only including configuration data nodes now. 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.
We define the IGMP model as a protocol-centric model, and the IGMP model augments "/rt:routing/rt:control-plane-protocols/ rt:control-plane-protocol" in [draft-acee-netmod-rfc8022bis-01] and would allow a single protocol instance per VRF.

augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
  +++rw igmp
    +++rw global
      +++rw enable?          boolean {global-admin-enable}?
      +++rw max-entries?     uint32 {global-max-entries}?
      +++rw max-groups?      uint32 {global-max-groups}?
      +++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
Internet-Draft          IGMP & MLD Yang Model             October 2017

3.2. MLD Configuration and Operational State

The MLD YANG model uses the same structure as IGMP YANG model. The
MLD module also defines the routing-control-plane-protocol-wide
configuration and operational state options separately in a three-
level hierarchy.

augment /rt:routing/rt:control-plane-protocols/rt:control-plane-protocol:
  +--rw mld
    +--rw global
      |  +--rw enable?        boolean {global-admin-enable}?
      |  +--rw max-entries?   uint32 {global-max-entries}?
      |  +--rw max-groups?    uint32 {global-max-groups}?
      |  +--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}?
    +-rw version?                      uint8
    +-rw join-group*                   inet:ipv6-address {intf-join-group}?
        +-rw source-addr    ssm-map-ipv6-addr-type
        +-rw group-policy   string

Liu & Guo, etc Expires April, 2018 [Page 7]
3.3. IGMP and MLD RPC

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

rpcs:

    +---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

<CODE BEGINS> file "ietf-igmp-mld@2017-10-20.yang"
module ietf-igmp-mld {
    namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld";
    // replace with IANA namespace when assigned
    prefix igmp-mld;

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

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

    import ietf-routing {
        prefix "rt";
    }

    import ietf-interfaces {
        prefix "if";
    }

    import ietf-ip {
        prefix ip;
    }

    organization
        "IETF PIM Working Group";

    contact
        "WG Web:  <http://tools.ietf.org/wg/pim/>";
        "WG List:  <mailto:pim@ietf.org>"
        "WG Chair: Stig Venaas"
            <mailto:stig@venaas.com>
        "WG Chair: Mike McBride"
            <mailto:mmcbride7@gmail.com>
        "Editor:  Xufeng Liu"
            <mailto:Xufeng_Liu@jabil.com>
        "Editor:  Feng Guo"
            <mailto:guofeng@huawei.com>
The module defines a collection of YANG definitions common 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 */

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
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 can join.";
  }

  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.";
    }
  }
} // global-state-attributes
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.";
    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;
default false;
description
  "Protocol packets should contain router alert IP option.";
}
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.";
}

} // 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.";
  }
}

grouping interfaces-config-attributes-mld {
  description "interfaces configuration for MLD.";
  uses interfaces-config-attributes;
  leaf version {
    type uint8 {
      range "1..2";
    }
  }

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 joins this multicast group on the interface.";
  }

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

  list static-group {
    if-feature intf-static-group;
    key "group-addr source-addr";
    description "A static multicast route, (*,G) or (S,G).";
  }
}
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 IGMPv2 or MLDv1 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";
        }
    }

leaf last-member-query-interval {
  type uint32 {
    range "1..4095";
  }
  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 group sources.";
}

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."
    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 "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 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.";
    }
    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;
    }
  }
}
leaf source-addr {
  type source-ipv6-addr-type;
  description
    "Multicast source IPv6 address.";
}

// interface-config-attributes-mld

grouping interface-config-attributes-mld {

description
  "Multicast group IPv6 address.";
}

leaf source-addr {
  type source-ipv6-addr-type;
  description
    "Multicast source IPv6 address.";
}

// 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 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
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";
  }
}

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" (}

Liu & Guo, etc Expires April, 2018 [Page 26]
description
    "In include mode";
}

enum "exclude" {
    description
    "In exclude mode.";
}

description
    "Filter mode for a multicast membership
    host may be either include or exclude.";
}

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
c 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
c 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/rt:control-plane-protocol"
{
  description
    "IGMP augmentation to routing control plane protocol
   configuration and state."

  container igmp {
    description
      "IGMP operational state data.";

    container global {
      description

Liu & Guo, etc Expires April, 2018 [Page 28]
"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
} // augment

augment "/rt:routing/rt:control-plane-protocols/rt:control-plane-protocol"
{ 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;
} // interface
} // interfaces
} // mld
} // augment

/*
 * RPCs
 */
rpc clear-igmp-groups {
  if-feature rpc-clear-groups;
  description
  "Clears the specified IGMP cache entries.";
  input {
    leaf interface {
      type string;
      description
      "Name of the IGMP interface.
      If it is not specified, groups from all interfaces are
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.";
    }
    leaf group {
      type inet:ipv6-address;
      description
        "Multicast group IPv6 address.
        If it is not specified, all MLD group entries are cleared.";
    }
    leaf source {
      type inet:ipv6-address;
      description
        "Multicast source IPv6 address.
        If it is not specified, all MLD source-group entries are cleared.";
    }
  }
} // rpc clear-mld-groups
5. Security Considerations

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

6. IANA Considerations

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

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

```
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
```

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

```
name:         ietf-igmp-mld
prefix:       igmp-mld
reference:    RFC XXXX
```

---
7. Acknowledgments

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

8. Contributing Authors

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

Email: liuyisong@huawei.com

9. References

9.1. Normative References


[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011


[draft-bjorklund-netmod-rfc7223bis-00] M. Bjorklund, "A YANG Data Model for Interface Management", draft-bjorklund-netmod-rfc7223bis-00, August 2017
9.2. Informative References


Authors’ Addresses

Xufeng Liu
Jabil
8281 Greensboro Drive, Suite 200
McLean VA 22102
USA
EMail: Xufeng_Liu@jabil.com

Feng Guo
Huawei
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China
Email: guofeng@huawei.com

Mahesh Sivakumar
Cisco Systems, Inc.
510 McCarthy Boulevard
Milpitas, California 95035
USA
Email: masivaku@cisco.com

Pete McAllister
Metaswitch Networks
100 Church Street
Enfield EN2 6BQ
UK
EMail: pete.mcallister@metaswitch.com

Anish Peter
Individual
EMail: anish.ietf@gmail.com
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.
carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

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

Table of Contents

1. Introduction ................................................ 2
   1.1. Terminology ............................................ 3
   1.2. Tree Diagrams .......................................... 3
   1.3. Prefixes in Data Node Names ............................. 3
2. Design of Data model ......................................... 4
   2.1. Scope of model ......................................... 4
   2.2. Optional capabilities ................................... 4
   2.3. Position of address family in hierarchy ............... 5
3. Module Structure ............................................ 5
   3.1. IGMP Configuration and Operational state ............... 5
   3.2. MLD Configuration and Operational State .............. 7
   3.3. IGMP and MLD RPC ....................................... 9
4. IGMP and MLD YANG Modules .................................... 9
5. Security Considerations ..................................... 33
6. IANA Considerations ........................................ 35
7. Acknowledgments ............................................ 36
8. Contributing Authors ........................................ 36
9. References ................................................. 36
   9.1. Normative References .................................. 36
   9.2. Informative References ................................. 38

1. Introduction

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

+-----------+--------------------+---------------------+
| Prefix    | YANG module        | Reference           |
+-----------+--------------------+---------------------+
    | yang      | ietf-yang-types    | [RFC6991]           |
    | inet      | ietf-inet-types    | [RFC6991]           |
    | if        | ietf-interfaces    | [RFC8343]           |
    | rt        | ietf-routing       | [RFC8349]           |

+-------------------------------------------------------------------+
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:
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
```
Internet-Draft IGMP & MLD Yang Model June 2018

```Yang
|     |  +--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}?
|     |  +--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}?
|     |     +--ro group* [group-address] inet:ipv6-address
```

Liu & Guo, etc Expires December, 2018 [Page 8]
3.3. IGMP and MLD RPC

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

rpcs:

---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

<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
description "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
 */
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.";
}

Liu & Guo, etc          Expires December, 2018                [Page 12]
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
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

Liu & Guo, etc Expires December, 2018 [Page 14]
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:
0 1 2 3 4 5 6 7
|1| exp | mant |
|++|+++|+++|
QQI = (mant | 0x10) << (exp + 3)
The maximum value of QQI is 31744.";
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.";

reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";

reference "RFC3376. Sec. 4.1.7, 8.2, 8.14.2.";

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";
    }
  }
}
# IGMP & MLD Yang Model

Internet-Draft

June 2018

characters are not allowed. ";

)}

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

\[ QQI = (mant | 0x10) \ll (exp + 3) \]
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.";
    }
}

Liu & Guo, etc Expires December, 2018 [Page 24]
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.";
    }

    uses interface-state-host-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.";
    }

    uses interface-state-host-attributes-igmp-mld;
}

Liu & Guo, etc Expires December, 2018 [Page 25]
"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;

Liu & Guo, etc       Expires December, 2018       [Page 26]
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;
    }
  }
}

Liu & Guo, etc Expires December, 2018 [Page 27]
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 {

Liu & Guo, etc Expires December, 2018 [Page 28]
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

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;
}
/*
 * RPCs
 */

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

// rpc clear-mld-groups

/*
 * Notifications
 */

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]:

Liu & Guo, etc Expires December, 2018 [Page 35]
7. Acknowledgments

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

8. Contributing Authors

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

Email: liuyisong@huawei.com

9. References

9.1. Normative References


9.2. Informative References


[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
Use of PIM Address List Hello across address families

draft-ietf-pim-ipv4-prefix-over-ipv6-nh-01.txt

Abstract

In the PIM Sparse Mode standard there is an Address List Hello option used to list secondary addresses of an interface. Usually the addresses would be of the same address family as the primary address. In this document we provide a use case for listing secondary addresses that are from a different family. In particular, 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). When using this combined with PIM, the Address List Hello option can be used to determine which PIM neighbor to use as RPF neighbor.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 30, 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
1. Introduction

The PIM Sparse Mode standard [RFC7761] defines an Address List Hello option used to list secondary addresses of an interface. It specifies that the addresses listed SHOULD be of the same address family as the primary address. It was not anticipated that it could be useful to list addresses of a different address family. This document describes a use-case for listing different address families.

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 progression of join and furthermore to build multicast tree. This document describes various approaches along with their use-cases and pros-cons.

Figure 1: Example Topology

```
+-------------+                   +-------------+
|             |                   |             |
| Router1    1::1/64          1::2/64 Router2 |
+----------------+           +--------I1---------+
| 1.1.1.1/24    1.1.1.2/24 |
+----------------+                   +             |
|             |                   |             |
+-------------+                   +-------------+
```

In example topology, Router1 and Router2 are PIMv4 and PIMv6 neighbors on Interface I1. Router2 learns prefix 10.1.1.1/32’s next-hop as 1::1/64 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, Router2 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, [RFC2119].
2. Solution

A PIM router can advertise its locally configured IPv6 addresses on the interface in PIMv4 Hello messages as per [RFC7761] section 4.3.4. Same applies for IPv4 address in PIMv6 Hello. PIM will keep this info for each neighbor in Neighbor-cache along with DR-priority, hold-time etc. Once IPv6 Next-hop is notified to PIMv4, it will look into neighbors on the notified RPF-interface and find PIMv4 neighbor advertising same IPv6 local address in secondary Neighbor-list. If such a match is found, that particular neighbor will be used as IPv4 RPF-Neighbor for initiating upstream join.

This method is valid for networks enabled with PIMv4 and PIMv6 both as well for the networks enabled with only PIMv4 with IPv6 BGP session or PIMv6 with IPv4 BGP session. This method doesn’t required 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

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

Email: ashutosh@avinetworks.com

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

Email: stig@cisco.com
Use of PIM Address List Hello across address families
draft-ietf-pim-ipv4-prefix-over-ipv6-nh-02.txt

Abstract

In the PIM Sparse Mode standard there is an Address List Hello option used to list secondary addresses of an interface. Usually the addresses would be of the same address family as the primary address. In this document we provide a use case for listing secondary addresses that are from a different family. In particular, 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). When using this combined with PIM, the Address List Hello option can be used to determine which PIM neighbor to use as RPF neighbor.

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 November 2, 2018.

Copyright Notice

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

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

The PIM Sparse Mode standard [RFC7761] defines an Address List Hello option used to list secondary addresses of an interface. It specifies that the addresses listed SHOULD be of the same address family as the primary address. It was not anticipated that it could be useful to list addresses of a different address family. This document describes a use-case for listing different address families.

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.

Figure 1: Example Topology

In example topology, Router1 and Router2 are PIMv4 and PIMv6 neighbors on Interface I1. Router2 learns prefix 10.1.1.1/32’s next-hop as 1::1/64 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, Router2 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, [RFC2119].
2. Solution

A PIM router can advertise its locally configured IPv6 addresses on the interface in PIMv4 Hello messages as per [RFC7761] section 4.3.4. Same applies for IPv4 address in PIMv6 Hello. PIM will keep this info for each neighbor in Neighbor-cache along with DR-priority, hold-time etc. Once IPv6 Next-hop is notified to PIMv4, it will look into neighbors on the notified RPF-interface and find PIMv4 neighbor advertising same IPv6 local address in secondary Neighbor-list. If such a match is found, that particular neighbor will be used as IPv4 RPF-Neighbor for initiating upstream join.

This method is valid for networks enabled with PIMv4 and PIMv6 both as well for the networks enabled with only PIMv4 with IPv6 BGP session or PIMv6 with IPv4 BGP session. This method doesn’t required 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

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

Email: ashutosh@avinetworks.com

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

Email: stig@cisco.com
BIER Flooding Mechanism

draft-zhang-bier-flooding-00.txt

Abstract

This document introduces a method to flood BIER information in hybrid network to build BIER forwarding plane.

Status of This Memo

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

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

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

This Internet-Draft will expire on April 3, 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 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.
1. Problem Statement

Some networks have been deployed widely are hybrid networks. There are different dynamic routing protocols running in the hybrid networks. Multicast services can also be provided in these kinds of networks because of the protocol independent feature of PIM.

BIER [I-D.ietf-bier-architecture] provides a new architecture for the forwarding of multicast data packets. It does not require a protocol for explicitly building multicast distribution trees, nor does it require intermediate nodes to maintain any per-flow state. [I-D.ietf-bier-isis-extensions] and [I-D.ietf-bier-ospf-bier-extensions] are good at establishing BIER forwarding plane in network which uses OSPF or IS-IS as BIER underlay protocol.
In the mentioned networks, there are more than one dynamic routing protocols running in the networks. For example in figure 1, this is a partial typical network in actually deployment. Two different dynamic routing protocols and are used in the network. Sometimes static configured routes also are used in some parts of the network. In order to deploy BIER multicast, we can divide the network into several BIER domains. Obviously the efficiency slows down due to multiple encapsulating/ decapsulating executions.

2. Solution

The Bootstrap Router mechanism (BSR) [RFC5059] is a commonly used mechanism for distributing dynamic Group to RP mappings in PIM. It is responsible for flooding information about such mappings throughout a PIM domain, so that all routers in the domain can have the same information. [I-D.ietf-pim-source-discovery-bsr] defines a mechanism that can flood any kind of information throughout a PIM domain. This document borrows the idea from the two drafts, introduces a mechanism to flood BIER node’s information throughout a
BIER domain to build BIER forwarding plane. Nodes can use unicast forwarding table directly to establish BIER forwarding plane.

The validation processing of PFM messages is the same as the definition in [I-D.ietf-pim-source-discovery-bsr] section 3.2.

BIER node originates BIER information TLV and optional associated sub-TLVs in PFM message. The PFM messages are flooded by throughout the BIER domain. BFR gets routing information from the unicast forwarding table directly, and computes BIER forwarding table. Then BIER forwarding plane is established.

2.1. Scheduled Update

Because PIM advertisement is scheduled, the node’s BIER information is refreshed periodically. In case one node’s BIER information changes or expires, the other nodes recompute the BIER forwarding table. The holdtime in the BIER information TLV is used to make the item expired.

2.2. Triggered Update

If the BIER node’s configuration changes, such as BFR-id, the node should send update PFM messages immediately. Then other nodes can recompute the new BIER forwarding table.

3. Message Format

3.1. PFM message

New TLVs are defined in PFM message to flood node’s BIER information, such as BFR-id, BFR-prefix and so on. The new TLVs align exactly with the definition and restrictions in [I-D.ietf-bier-isis-extensions] and [I-D.ietf-bier-ospf-bier-extensions].
The format of PFM message is defined in [I-D.ietf-pim-source-discovery-bsr].

Originator Address: The router’s address that originate the message. The address SHOULD be the same with the node’s BFR-prefix.

The other fields is the same as definition in [I-D.ietf-pim-source-discovery-bsr].

3.2. BIER information TLV

A new type of TLV is defined in PFM message. This new type TLV is named by BIER information TLV. Two types of sub-TLV are associated with it. There is no optional BIER tree type sub-TLV in PFM message because of the independence of routing protocol.
o Type: The value of type should be assigned by IANA.

o Length: The total length of the BIER information TLV except for the first two fields.

o Reserved: Must be 0 on transmission, ignored on reception. May be used in future version. 1 octets.

o Subdomain-id: Unique value identifying the BIER sub-domain. 1 octet.

o BFR-id: The value of BFR-id defined in [BIER-arch], 2 octets. 0 is invalid value. If the value of this field is set to 0, the whole TLV MUST be ignored and not forwarded.

o Holdtime: The life cycle of the BIER information. The default value is 60s.

o BFR-prefix: The BFR-prefix of the node in this sub-domain. The format for this address is given in the Encoded-Unicast address in [RFC7761].

A node may belong to several BIER sub-domains, so it is possible that there are multiple BIER information TLVs in the FFM message.

3.3. BIER MPLS Encapsulation sub-TLV

In case the nodes in the network support MPLS forwarding, BIER MPLS encapsulation sub-TLV can be advertised for a specific bitstring length for a certain (MT,subdomain). This sub-TLV may appear multiple times within single BIER information TLV. The format and restriction is the same as the definition in [I-D.ietf-bier-isis-extensions] and [I-D.ietf-bier-ospf-bier-extensions].
The type value of this sub-TLV should be assigned by IANA. The suggestion value is 1.

3.4. Optional BIER sub-domain BSL conversion sub-TLV

The format and restriction is the same as the definition in [I-D.ietf-bier-isis-extensions] and [I-D.ietf-bier-ospf-bier-extensions]. The type value of this sub-TLV should be assigned by IANA. The suggestion value is 2.

4. Security Considerations

The security considerations are mainly similar to what is documented in [I-D.ietf-pim-source-discovery-bsr].

5. IANA Considerations

This document requires the assignment of a new PFM TLV type for the BIER information Flooding Mechanism. IANA is also requested to create two sub-TLV types for BIER MPLS encapsulation sub-TLV and BIER sub-domain BSL conversion sub-TLV.

6. Normative References

[I-D.ietf-bier-architecture]

[I-D.ietf-bier-isis-extensions]

[I-D.ietf-bier-ospf-bier-extensions]

[I-D.ietf-pim-source-discovery-bsr]


Authors’ Addresses

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

Email: zhang.zheng@zte.com.cn

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

Email: xu.benchong@zte.com.cn
A Yang Data Model for IGMP and MLD Snooping
draft-zhao-pim-igmp-mld-snooping-yang-03.txt

Abstract

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

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}?
|     +--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*     l2vpn-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*            l2vpn-instance-ac-ref
|         |     +--rw vpls-outgoing-pw*            l2vpn-instance-pw-ref
|         +--ro entries-count?                     uint32
|         +--ro bridge-mrouter-interface*        if:interface-ref
|         +--ro vpls-mrouter-interface*          l2vpn-instance-pw-ref
|         +--ro group* [address]
|         |     +--ro address                    inet:ipv4-address
```yang
++--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* 12vpn-instance-ac-ref
  ++--ro vpls-outgoing-pw* 12vpn-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* l2vpn-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* l2vpn-instance-ac-ref 
| | +--rw vpls-outgoing-pw* l2vpn-instance-pw-ref 
| +--ro entries-count? uint32 
| +--ro bridge-mrouter-interface* if:interface-ref 
| +--ro vpls-mrouter-interface* l2vpn-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* l2vpn-instance-ac-ref
Internet-Draft      IGMP & MLD Snooping Yang Model         October 2017

|     |     +--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:ipv6-address
|     |     +--ro host* [host-address] {explicit-tracking}?
|     |        +--ro host-address        inet:ipv6-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
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

Zhao                        Expires April 26, 2018          [Page 10]
Internet-Draft      IGMP & MLD Snooping Yang Model         October 2017

|                        +--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)
| | +--rw primary-ac
| |     +--rw name? string
ce-ref
| |     +--rw igmp-snooping-instance? igmp-snooping-instance-ref
e-ref
| |     +--rw mld-snooping-instance? mld-snooping-instance-ref
e-ref
| +--:(primary-pw)
|     +--rw primary-pw* [name]
|         +--rw name string
ce-ref
|         +--rw igmp-snooping-instance? igmp-snooping-instance-ref
e-ref
|     +--rw mld-snooping-instance? mld-snooping-instance-ref
e-ref

++-rw (backup)?
   +--:(backup-ac)
   |     +--rw backup-ac
   |         +--rw name? string
ce-ref
   |         +--rw igmp-snooping-instance? igmp-snooping-instance-ref
e-ref
   |     +--rw mld-snooping-instance? mld-snooping-instance-ref
e-ref
   | +--:(backup-pw)
   |     +--rw backup-pw* [name]
   |         +--rw name string
ce-ref
   |         +--rw igmp-snooping-instance? igmp-snooping-instance-ref
e-ref
   |     +--rw mld-snooping-instance? mld-snooping-instance-ref
e-ref
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
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

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

typedef source-ipv6-addr-type {
  type union {
    type enumeration {
      enum '*' {
        description
        "Any source address.";
      }
    }
    type inet:ipv6-address;
  } // 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 forwarding";
        }
        leaf-list vpls-outgoing-ac {
            when "ims:type = 'vpls'";
            type l2vpn-instance-ac-ref;
            description "Outgoing ac in vpls forwarding";
        }
    }
}

Zhao                    Expires April 26, 2018                [Page 19]
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-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;
    }

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-mls-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;
  }

  list static-vpls-multicast-group {
    if-feature static-vpls-multicast-group;
  }

  list static-bridge-multicast-group {
    if-feature static-bridge-multicast-group;
  }

  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";
  }

  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";
  }
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";
    }

Zhao                    Expires April 26, 2018                [Page 27]
grouping instance-state-attributes-igmp-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;

        }
    }
} // instance-config-attributes-igmp-mld-snooping
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.";
  }
} // general-statistics-sent-received
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";
                }
            }
            description "An L2VPN instance’s " +
            "attachment circuit list";
        }
        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;
                }
            }
        }
    }
}

Zhao                    Expires April 26, 2018                [Page 32]
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";
}

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

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.";
    }
  }
}
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 {

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

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.";
      }
      container igmp-snooping-instance {
        type igmp-snooping-instance-ref;
        description "Configure igmp-snooping instance under the vlan view";
      }
      container 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";
  }
}
} //interfaces
} //vlan
} //bridge-vlan
} //component
} //bridge
} //bridges

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 EVPN 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 EVPN 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.";
    }
  }
} // rpc clear-mld-snooping-groups
"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

the Network Configuration Protocol (NETCONF)", RFC 6020,
October 2010.

October 2010.

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


[RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast
Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.


[draft-bjorklund-netmod-rfc7223bis-00] M. Bjorklund, "A YANG Data Model for Interface Management", draft-bjorklund-netmod-rfc7223bis-00, August 21, 2017


Authors’ Addresses

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

Email: hongji.zhao@ericsson.com

Xufeng Liu
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