

PIM WG
Internet-Draft
Intended status: Standards Track
Expires: December 8, 2017

Zheng. Zhang
Fangwei. Hu
BenChong. Xu
ZTE Corporation
Mankamana. Prasad Mishra
Cisco Systems
June 6, 2017

PIM DR Improvement
draft-ietf-pim-dr-improvement-03.txt

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.



Figure 1: An example of multicast network

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.

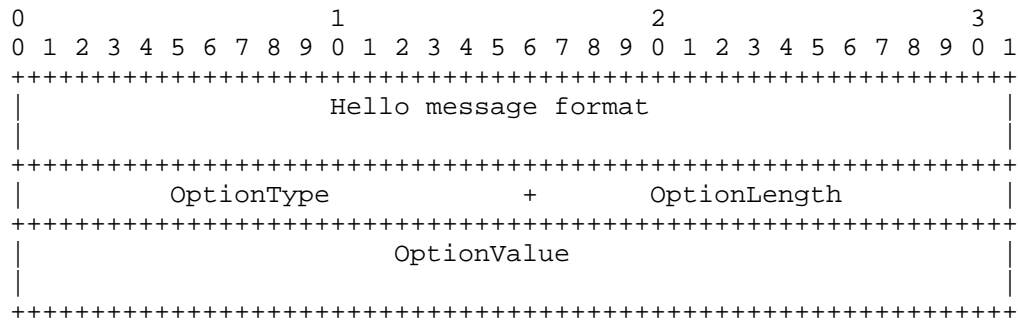


Figure 2: Hello message format

3.1. DR Address Option format

- o OptionType : The value is TBD.
- o OptionLength: If the network is support IPv4, the OptionLength is 4 octets. If the network is support IPv6, the OptionLength is 16 octets.
- o 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

- o OptionType : The value is TBD.
- o OptionLength: If the network is support IPv4, the OptionLength is 4 octets. If the network is support IPv6, the OptionLength is 16 octets.
- o 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 receive hello messages from other routers, the router send hello messages with the value of DR is set to the IP address of 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 receive 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.

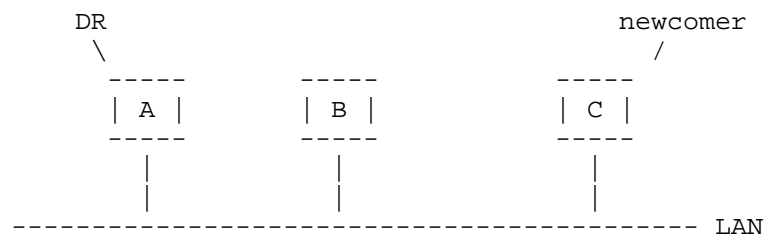


Figure 3

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 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 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

- [HRW] IEEE, "Using name-based mappings to increase hit rates", IEEE HRW, February 1998.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <<http://www.rfc-editor.org/info/rfc2328>>.
- [RFC2362] Estrin, D., Farinacci, D., Helmy, A., Thaler, D., Deering, S., Handley, M., Jacobson, V., Liu, C., Sharma, P., and L. Wei, "Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification", RFC 2362, DOI 10.17487/RFC2362, June 1998, <<http://www.rfc-editor.org/info/rfc2362>>.
- [RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <<http://www.rfc-editor.org/info/rfc4601>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<http://www.rfc-editor.org/info/rfc7761>>.

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 District
Nanjing
China

Email: xu.benchong@zte.com.cn

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

Email: mankamis@cisco.com

PIM WG
Internet-Draft
Intended status: Standards Track
Expires: 8 September 2022

Z. Zhang
ZTE Corporation
F. Hu
Individual
B. Xu
ZTE Corporation
M. Mishra
Cisco Systems
7 March 2022

Protocol Independent Multicast - Sparse Mode (PIM-SM) Designated Router
(DR) Improvement
draft-ietf-pim-dr-improvement-13

Abstract

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

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

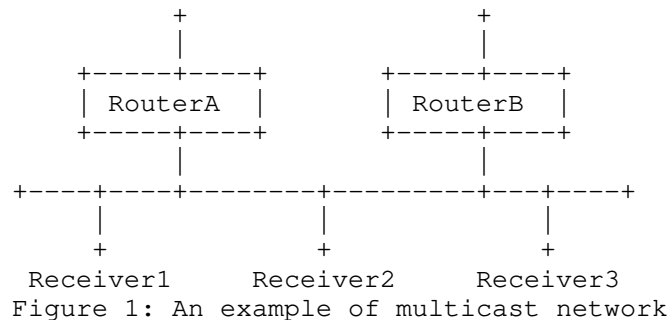
Table of Contents

1. Introduction	2
1.1. Keywords	3
2. Terminology	3
3. Protocol Specification	4
3.1. Election Algorithm	5
3.2. Sending Hello Messages	7
3.3. Receiving Hello Messages	8
3.4. Working with the DRLB function	9
4. PIM Hello message format	9
4.1. DR Address Option format	9
4.2. BDR Address Option format	10
4.3. Error handling	10
5. Backwards Compatibility	10
6. Security Considerations	11
7. IANA Considerations	11
8. Acknowledgements	12
9. References	12
9.1. Normative References	12
9.2. Informative References	12
Authors' Addresses	13

1. Introduction

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

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



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

(a) Both routers are on the network, and RouterB is elected as the DR. If RouterB then fails, multicast packets are discarded until RouterA is elected as DR, and it assumes the multicast flows on the LAN. As detailed in [RFC7761], a DR's election is triggered after the current DR's Hello_Holdtime expires. The failure detection and election procedures may take several seconds. That is too long for modern multicast services.

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

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

1.1. Keywords

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

2. Terminology

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

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

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

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

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

3. Protocol Specification

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

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

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

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

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

If the DR becomes unreachable on the LAN, the BDR MUST take over all the DR functions, including multicast flow forwarding and sending the Register messages. Mechanisms outside the scope of this specification, such as [RFC9186] or BFD Asynchronous mode [RFC5880] can be used for faster failure detection.

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

3.1. Election Algorithm

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

- * The DR is elected among the DR candidates directly. If there is no DR candidates, i.e., all the routers advertise the DR Address options with zero OptionValue, the elected BDR will be the DR. And then the BDR is elected again from the other routers in the LAN.
- * The BDR election is not sticky. Whatever there is a router that advertise the BDR Address option, the router which has the highest priority, expect for the elected DR, is elected as the BDR. That is the BDR may be the router which has the highest priority in the LAN.
- * The advertisement is through PIM Hello message.

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

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

The pseudocode is shown below: A BDR election function is added, and the DR function is updated. The validneighbor function means that a valid Hello message has been received from this neighbor.


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

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

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

- * The router, that can be elected as DR, has the highest priority among the DR candidates. The elected DR may not be the one that has the highest priority in the LAN.
- * The router that supports the election algorithm defined in section 3.1 MUST advertise the DR Address option defined in section 4.1 in PIM Hello message, and SHOULD advertise the BDR Address option defined in section 4.2 in PIM Hello message. In case a DR is elected and no BDR is elected, only the DR Address option is advertised in the LAN.

3.2. Sending Hello Messages

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

After the election, if there is one existed DR in the LAN, the DR remains unchanged. If there is no existed DR in the LAN, a new DR is elected, the routers in the LAN MUST send the Hello message with the OptionValue of DR Address option set to the elected DR. If there are more than one routers with non-zero DR priority in the LAN, a BDR is also elected. Then the routers in the LAN MUST send the Hello message with the OptionValue of BDR Address option set to the elected BDR. Any DROther router MUST NOT use its IP addresses in the DR/BDR Address option.

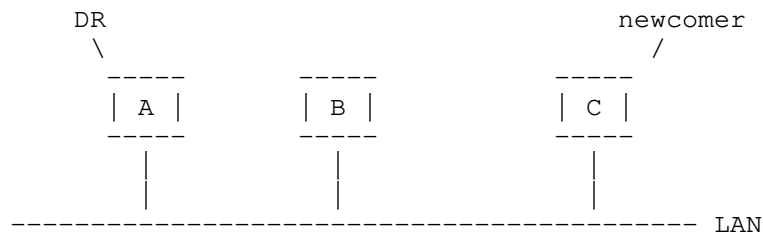


Figure 2

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

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

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

3.3. Receiving Hello Messages

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

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

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

3.4. Working with the DRLB function

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

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

4. PIM Hello message format

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

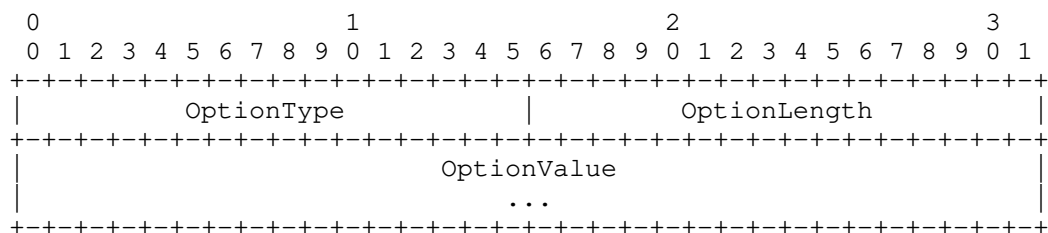


Figure 3: Hello Option Format

4.1. DR Address Option format

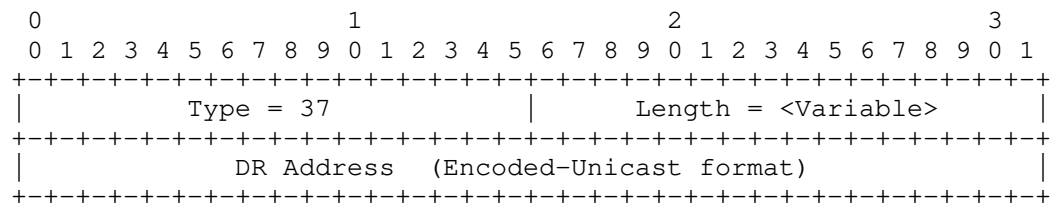


Figure 4: DR Address Option

* OptionType : The value is 37.

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

- * DR Address: If the IP version of the PIM message is IPv4, the value MUST be the IPv4 address of the DR. If the IP version of the PIM message is IPv6, the value MUST be the link-local address of the DR.

4.2. BDR Address Option format

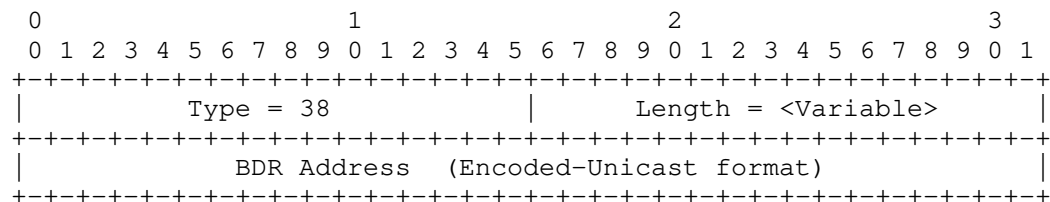


Figure 5: BDR Address Option

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

4.3. Error handling

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

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

5. Backwards Compatibility

Any router using the DR and BDR Address Options MUST set the corresponding OptionValues. If at least one router on a LAN doesn't send a Hello message, including the DR Address Option, then the specification in this document MUST NOT be used. For example, the routers in a LAN all support the options defined in this document, the DR/BDR is elected. A new router which doesn't support the options joins, when the hello message without DR Address Option is received, all the router MUST switch the election function back

immediately. This action results in all routers using the DR election function defined in [RFC7761] or [I-D.ietf-pim-bdr]. Both this draft and the draft [I-D.ietf-pim-bdr], introduce a backup DR. The later draft does this without introducing new options but does not consider the sticky behavior. In case there is router which doesn't support the DR/BDR Address Option defined in this document, the routers SHOULD take the function defined in [I-D.ietf-pim-bdr] if all the routers support it, otherwise the router SHOULD use the function defined in [RFC7761].

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

6. Security Considerations

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

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

7. IANA Considerations

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

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

Table 1

8. Acknowledgements

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

9. References

9.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2328] Moy, J., "OSPF Version 2", STD 54, RFC 2328, DOI 10.17487/RFC2328, April 1998, <<https://www.rfc-editor.org/info/rfc2328>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8775] Cai, Y., Ou, H., Vallepalli, S., Mishra, M., Venaas, S., and A. Green, "PIM Designated Router Load Balancing", RFC 8775, DOI 10.17487/RFC8775, April 2020, <<https://www.rfc-editor.org/info/rfc8775>>.

9.2. Informative References

- [I-D.ietf-pim-bdr] Mishra, M., Santhanam, S., Paramasivam, A., Goh, J., and G. S. Mishra, "PIM Backup Designated Router Procedure", Work in Progress, Internet-Draft, draft-ietf-pim-bdr-00, 10 September 2021, <<https://www.ietf.org/archive/id/draft-ietf-pim-bdr-00.txt>>.
- [RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <<https://www.rfc-editor.org/info/rfc5880>>.

[RFC9186] Mirsky, G. and X. Ji, "Fast Failover in Protocol Independent Multicast - Sparse Mode (PIM-SM) Using Bidirectional Forwarding Detection (BFD) for Multipoint Networks", RFC 9186, DOI 10.17487/RFC9186, January 2022, <<https://www.rfc-editor.org/info/rfc9186>>.

Authors' Addresses

Zheng (Sandy) Zhang
ZTE Corporation
No. 50 Software Ave, Yuhuatai District
Nanjing
China
Email: zhang.zheng@zte.com.cn

Fangwei Hu
Individual
Shanghai
China
Email: hufwei@gmail.com

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

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

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 30, 2017

Yiqun. Cai
Heidi. Ou
Alibaba Group
Sri. Vallepalli
Mankamana. Mishra
Stig. Venaas
Cisco Systems
Andy. Green
British Telecom
June 28, 2017

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.

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	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. Protocol Specification	11
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:

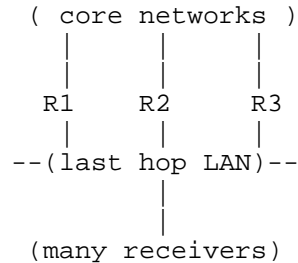


Figure 1: Last Hop LAN

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.

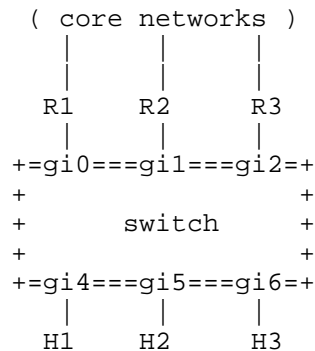


Figure 2: Last Hop Network with Ethernet Switch

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 1Gbps. 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

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.

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:

- o 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.
- o 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) PIMHello 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,

- o RP Hash Mask
- o Source Hash Mask
- o 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).

- o 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.
- o 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.
- o 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 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 as an example, with detailed description on hashvalue_RP.

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

$$\text{hashvalue_RP} = (((\text{RP_address} \& \text{RP_hashmask}) \gg N) \& 0\text{xFFFF}) \% M$$

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:

$$(((192.0.2.1 \& 0.0.255.0) \gg 8) \& 0xFFFF \% 3) = 2 \% 3 = 2$$

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:

$$(((198.51.100.2 \& 0.0.255.0) \gg 8) \& 0xFFFF \% 3) = 100 \% 3 = 1$$

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

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

$$\text{hashvalue_Group} = (((\text{Group_address} \& \text{Group_hashmask}) \gg N) \& 0xFFFF) \% M$$

Compare hashvalue_{Group} with Ordinal number assigned to Router X, to decide if Router X is the GDR.

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

$$\text{hashvalue_SG} = (((\text{Source_address} \& \text{Source_hashmask}) \gg N_S) \& 0xFFFF) \wedge (((\text{Group_address} \& \text{Group_hashmask}) \gg N_G) \& 0xFFFF) \% M$$

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

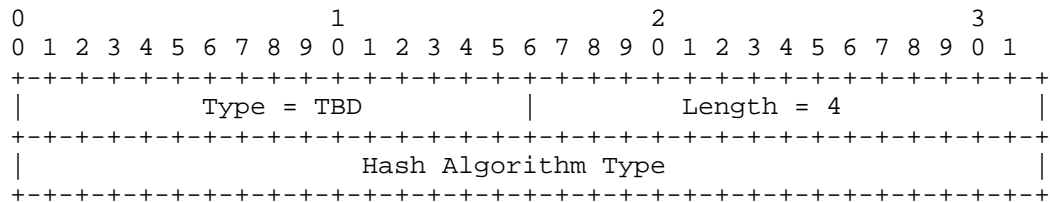


Figure 3: Capability Hello Option

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

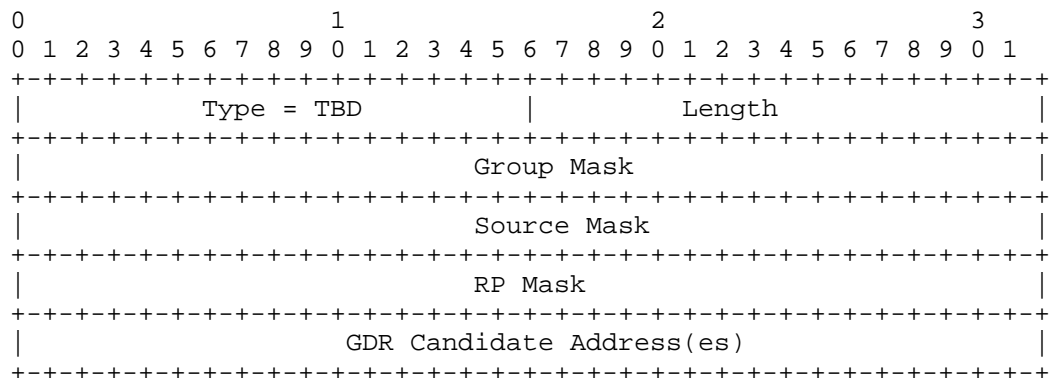


Figure 4: GDR Hello Option

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. Protocol Specification

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, DRLBGRD Hello option would be received from the current PIM DR on link.

A GDR Candidate may receive a DRLBGRD 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 DRLBGRD Hello Option from a PIM router which is not DR. The GDR Candidate MUST ignore such DRLBGRD 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 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 hased as GDR , it need to build multicast forwarding tree.

3. If a router receives IGMP/MLD report for flow 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 trees, 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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 4291, DOI 10.17487/RFC4291, February 2006, <<http://www.rfc-editor.org/info/rfc4291>>.
- [RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <<http://www.rfc-editor.org/info/rfc4601>>.
- [RFC6395] Gulrajani, S. and S. Venaas, "An Interface Identifier (ID) Hello Option for PIM", RFC 6395, DOI 10.17487/RFC6395, October 2011, <<http://www.rfc-editor.org/info/rfc6395>>.

12.2. Informative References

[HELLO-OPT]

IANA, "PIM Hello Options", IANA PIM-HELLO-OPTIONS, March 2007.

Authors' Addresses

Yiqun Cai
Alibaba Group

Email: yiqun.cai@alibaba-inc.com

Heidi Ou
Alibaba Group

Sri Vallepalli
Cisco Systems
3625 Cisco Way,
San Jose, 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

Andy Green
British Telecom
Adastral Park
Ipswich IP5 2RE
United Kingdom

Email: andy.da.green@bt.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 6, 2020

Y. Cai
H. Ou
Alibaba Group
S. Vallepalli
M. Mishra
S. Venaas
Cisco Systems, Inc.
A. Green
British Telecom
January 3, 2020

PIM Designated Router Load Balancing
draft-ietf-pim-drlb-15

Abstract

On a multi-access network, one of the PIM-SM (PIM Sparse Mode) routers is elected as a Designated Router. One of the responsibilities of the Designated Router is to track local multicast listeners and forward data to these listeners if the group is operating in PIM-SM. This document specifies a modification to the PIM-SM protocol that allows more than one of the PIM-SM routers to take on this responsibility so that the forwarding load can be distributed among multiple 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 July 6, 2020.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Terminology	5
3. Applicability	5
4. Functional Overview	5
4.1. GDR Candidates	6
5. Protocol Specification	7
5.1. Hash Mask and Hash Algorithm	7
5.2. Modulo Hash Algorithm	8
5.2.1. Modulo Hash Algorithm Examples	9
5.2.2. Limitations	10
5.3. PIM Hello Options	11
5.3.1. PIM DR Load Balancing Capability (DRLB-Cap) Hello Option	11
5.3.2. PIM DR Load Balancing List (DRLB-List) Hello Option	12
5.4. PIM DR Operation	13
5.5. PIM GDR Candidate Operation	14
5.6. DRLB-List Hello Option Processing	14
5.7. PIM Assert Modification	15
5.8. Backward Compatibility	16
6. Operational Considerations	16
7. IANA Considerations	17
7.1. Initial registry	17
7.2. Assignment of new Hash Algorithms	17
8. Security Considerations	17
9. Acknowledgement	18
10. References	18
10.1. Normative References	18
10.2. Informative References	19
Authors' Addresses	19

1. Introduction

On a multi-access LAN, such as an Ethernet, with one or more PIM-SM (PIM Sparse Mode) [RFC7761] routers, one of the PIM-SM routers is elected as a Designated Router (DR). The PIM DR has two responsibilities in the PIM-SM protocol. For any active sources on a

LAN, the PIM DR is responsible for registering with the Rendezvous Point (RP) if the group is operating in PIM-SM. Also, the PIM DR is responsible for tracking local multicast listeners and forwarding to these listeners if the group is operating in PIM-SM.

Consider the following LAN in Figure 1:

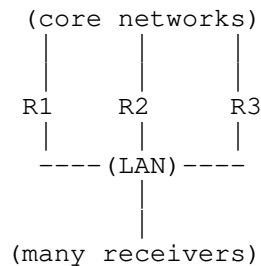


Figure 1: LAN with receivers

Assume R1 is elected as the DR. According to the PIM-SM protocol, R1 will be responsible for forwarding traffic to that LAN on behalf of all local members. In addition to keeping track of membership reports, R1 is also responsible for initiating the creation of source and/or shared trees towards the senders or the RPs. The membership reports would be IGMP or MLD messages. This applies to any versions of the IGMP and MLD protocols. The most recent versions are IGMPv3 [RFC3376] and MLDv2 [RFC3810].

Having a single router acting as DR and being responsible for data plane forwarding leads to several issues. One of the issues is that the aggregated bandwidth will be limited to what R1 can handle with regards to capacity of incoming links, the interface on the LAN, and total forwarding capacity. It is very common that a LAN consists of switches that run IGMP/MLD or PIM snooping [RFC4541]. This allows the forwarding of multicast packets to be restricted only to segments leading to receivers that 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.

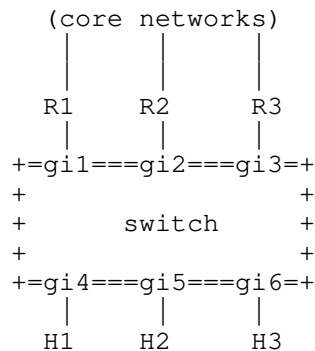


Figure 2: LAN with Ethernet Switch

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 gi1, can only handle a throughput of 1Gbps. And if R1 is the only DR (the PIM DR elected using the procedure defined by [RFC7761]) 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 gi1. In other words, the entire network's throughput is limited by the single connection between the PIM DR and the switch (or LAN as in Figure 1).

Another important issue is related to failover. If R1 is the only forwarder on a 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 were 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.

This document specifies 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", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

With respect to PIM-SM, this document follows the terminology that has been defined in [RFC7761].

This document also introduces the following new acronyms:

- o GDR: Group Designated Router. For each multicast flow, either a (*,G) for Any-Source Multicast (ASM), or an (S,G) for Source-Specific Multicast (SSM) [RFC4607], 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.
- o GDR Candidate: a router that has the potential to become a GDR. There might be multiple GDR Candidates on a LAN, but only one can become the GDR for a specific multicast flow.

3. Applicability

The extension specified in this document applies to PIM-SM routers acting as last hop routers (there are directly connected receivers). It does not alter the behavior of a PIM DR, or any other routers, on the first hop network (directly connected sources). 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 PIM 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 source registration does not justify the additional complexity required to support it.

4. Functional Overview

In the PIM DR election as defined in [RFC7761], when multiple routers are connected to a multi-access LAN (for example, an Ethernet), one of them is elected 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 own 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 [RFC7761].

In order to share forwarding load among last hop routers, besides the normal PIM DR election, one or more GDRs are elected on the 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 [Section 5.1] is used to select one of the routers to be the GDR. The new DR Load Balancing Capability (DRLB-Cap) PIM Hello Option is used to announce the Capability as well as the Hash Algorithm type. Routers with the new DRLB-Cap Option advertised in their PIM Hello, 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 Candidate Addresses, are announced by the DR in a new DR Load Balancing List (DRLB-List) 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. 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 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, each announcing a DRLB-Cap option. 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 advertise the same Hash Algorithm for GDR election, while R3 advertises 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 DRLB-List Hello Option.

5. Protocol Specification

5.1. Hash Mask and Hash Algorithm

A Hash Mask is used to extract a number of bits from the corresponding IP address field (32 for IPv4, 128 for IPv6) and calculate a hash value. A hash value is used to select a GDR from GDR Candidates advertised by the PIM DR. Hash masks allow for certain flows to always be forwarded by the same GDR, by ignoring certain bits in the hash value calculation, so that the hash values are the same. For example, 0.0.255.0 defines a Hash Mask for an IPv4 address that masks the first, the second, and the fourth octets, which means that only the third octet will influence the hash value computed. Note that the masks need not be a contiguous set of bits. E.g, for IPv4, 15.15.15.15 would be a valid mask.

In the text below, a hash mask is in some places said to be zero. A hash mask is zero if no bits are set. That is, 0.0.0.0 for IPv4 and :: for IPv6. Also, a hash mask is said to be an all-bits-set mask if it is 255.255.255.255 for IPv4 or ffff:ffff:ffff:ffff:ffff:ffff:ffff:ffff for IPv6.

There are three Hash Masks defined:

- o RP Hash Mask
- o Source Hash Mask
- o 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 values. An implementation SHOULD have default hash mask values as follows. The default RP Hash Mask SHOULD be zero (no bits set). The default Source and Group Hash Masks SHOULD both be all-bits-set masks. These default values are likely acceptable for most deployments, and simplify configuration. There is only a need to use other masks if one needs to ensure that certain flows are forwarded by the same GDR.

The DRLB-List Hello Option contains a list of GDR Candidates. The first one listed has ordinal number 0, the second listed ordinal number 1, and the last one has ordinal number N - 1 if there are N candidates listed. The hash value computed will be the ordinal

number of the GDR Candidate that is acting as GDR for the flow in question.

The input to be hashed is determined as follows:

- o If the group is in ASM mode and the RP Hash Mask announced by the PIM DR is not zero (at least one bit is set), calculate the value of `hashvalue_RP` [Section 5.2] to determine the GDR.
- o If the group is in ASM mode and the RP Hash Mask announced by the PIM DR is zero (no bits are set), obtain the value of `hashvalue_Group` [Section 5.2] to determine the GDR.
- o If the group is in SSM mode, use `hashvalue_SG` [Section 5.2] to determine the GDR.

A simple Modulo Hash Algorithm is defined in this document. However, to allow another Hash Algorithms to be used, a 1-octet "Hash Algorithm" field is included in the DRLB-Cap Hello Option to specify the Hash Algorithm used by the router.

If different Hash Algorithms are advertised among the routers on a LAN, only the routers advertising the same Hash Algorithm as the DR (as well as having the same DR priority as the DR) are eligible for GDR election.

5.2. Modulo Hash Algorithm

As part of computing the hash, the notation `LSZC(hash_mask)` is used to denote the number of zeroes counted from the least significant bit of a Hash Mask `hash_mask`. As an example, `LSZC(255.255.128)` is 7 and also `LSZC(ffff:8000::)` is 111. If all bits are set, `LSZC` will be 0. If the mask is zero, then `LSZC` will be 32 for IPv4, and 128 for IPv6.

The number of GDR Candidates is denoted as `GDRC`.

The idea behind the Modulo Hash Algorithm is in simple terms that the corresponding mask is applied to a value, then the result is shifted right `LSZC(mask)` bits so that the least significant bits that were masked out are not considered. Then this result is masked by `0xffffffff`, keeping only the last 32 bits of the result (this only makes a difference for IPv6). Finally, the hash value is this result modulo the number of GDR Candidates (`GDRC`).

The Modulo Hash Algorithm for computing the values `hashvalue_RP`, `hashvalue_Group` and `hashvalue_SG` is defined as follows.

`hashvalue_RP` is calculated as:

$$(((RP_address \& RP_mask) \gg LSZC(RP_mask)) \& 0xffffffff) \% GDRC$$

RP_address is the address of the RP defined for the group and
RP_mask is the RP Hash Mask.

hashvalue_Group is calculated as:

$$(((Group_address \& Group_mask) \gg LSZC(Group_mask)) \& 0xffffffff) \% GDRC$$

Group_address is the group address and Group_mask is the Group Hash Mask.

hashvalue_SG is calculated as:

$$((((Source_address \& Source_mask) \gg LSZC(Source_mask)) \& 0xffffffff) \wedge (((Group_address \& Group_mask) \gg LSZC(Group_mask)) \& 0xffffffff)) \% GDRC$$

Group_address is the group address and Group_mask is the Group Hash Mask.

5.2.1. Modulo Hash Algorithm Examples

To help illustrate the algorithm, consider this example. Router X with IPv4 address 203.0.113.1 receives a DRLB-List 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:

LSZC(0.0.255.0) is 8 and GDRC is 3. The hashvalue_RP for Group1 with RP RP1 is:

$$(((192.0.2.1 \& 0.0.255.0) \gg 8) \& 0xffffffff \% 3) = 2 \% 3 = 2$$

which matches the ordinal number assigned to Router X. Router X will be the GDR for Group1.

The hashvalue_RP for Group2 with RP RP2 is:

$$(((198.51.100.2 \& 0.0.255.0) \gg 8) \& 0xffffffff \% 3) = 100 \% 3 = 1$$

which is different from the ordinal number of Router X (2). Hence, Router X will not be GDR for Group2.

For IPv6 consider this example, similar to the above. Router X with IPv6 address fe80::1 receives a DRLB-List Hello Option from the DR, which announces RP Hash Mask ::ffff:ffff:ffff:0 and a list of GDR Candidates, sorted by IP addresses from high to low: fe80::3, fe80::2 and fe80::1. The ordinal number assigned to those addresses would be:

0 for fe80::3; 1 for fe80::2; 2 for fe80::1 (Router X).

Assume there are 2 RPs: RP1 2001:db8::1:0:5678:1 for Group1 and RP2 2001:db8::1:0:1234:2 for Group2. Following the modulo Hash Algorithm:

LSZC(::ffff:ffff:ffff:0) is 16 and GDRC is 3. The hashvalue_RP for Group1 with RP RP1 is:

$$(((2001:db8::1:0:5678:1 \& ::ffff:ffff:ffff:0) \gg 16) \& 0xffffffff \% 3) = ((::1:0:5678:0 \gg 16) \& 0xffffffff \% 3) = (::1:0:5678 \& 0xffffffff \% 3) = ::5678 \% 3 = 2$$

which matches the ordinal number assigned to Router X. Router X will be the GDR for Group1.

The hashvalue_RP for Group2 with RP RP2 is:

$$(((2001:db8::1:0:1234:1 \& ::ffff:ffff:ffff:0) \gg 16) \& 0xffffffff \% 3) = ((::1:0:1234:0 \gg 16) \& 0xffffffff \% 3) = (::1:0:1234 \& 0xffffffff \% 3) = ::1234 \% 3 = 1$$

which is different from the ordinal number of Router X (2). Hence, Router X will not be GDR for Group2.

5.2.2. Limitations

The Modulo Hash Algorithm has poor failover characteristics when a shared LAN has more than two GDRs. In the case of more than two GDRs on a LAN, when one GDR fails, all of the groups may be reassigned to a different GDR, even if they were not assigned to the failed GDR. However, many deployments use only two routers on a shared LAN for redundancy purposes. Future work may define new Hash Algorithms where only groups assigned to the failed GDR get reassigned.

The Modulo Hash Algorithm will use at most 32 consecutive bits of the input addresses for its computation. Exactly which bits are used of the source, group or RP addresses, depend on the respective masks.

This limitation may be an issue for IPv6 deployments, since not all bits of the IPv6 addresses are considered. If this causes operational issues, a new hash algorithm would need to be defined.

5.3. PIM Hello Options

PIM routers include a new option, called "Load Balancing Capability (DRLB-Cap)" in their PIM Hello messages.

Besides this DRLB-Cap Hello Option, the elected PIM DR also includes a new "DR Load Balancing List (DRLB-List) Hello Option". The DRLB-List Hello Option consists of three Hash Masks as defined above and also a list of GDR Candidate addresses on the LAN. It is recommended that the GDR Candidate addresses are sorted in descending order. This ensures that when using algorithms such as the Modulo algorithm in this document, that it is predictable which GDR is responsible for which groups, regardless of the order the DR learned about the candidates.

5.3.1. PIM DR Load Balancing Capability (DRLB-Cap) Hello Option

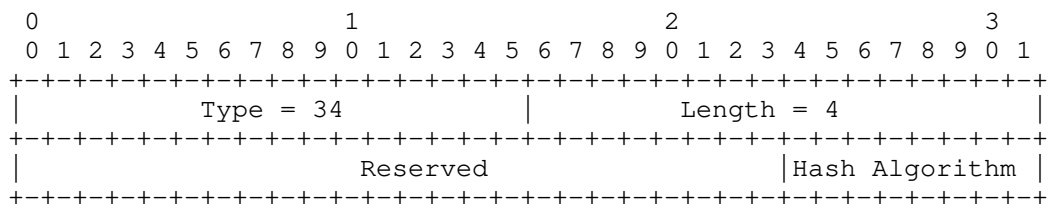


Figure 3: PIM DR Load Balancing Capability Hello Option

Type: 34

Length: 4

Reserved: Transmitted as zero, ignored on receipt.

Hash Algorithm: Hash Algorithm type. A value listed in the IANA Designated Router Load Balancing Hash Algorithms registry. 0 is used for the Modulo algorithm defined in this document.

This DRLB-Cap Hello Option MUST be advertised by routers on all interfaces where DR Load Balancing is enabled. Note that the option is included at most once.

5.3.2. PIM DR Load Balancing List (DRLB-List) Hello Option

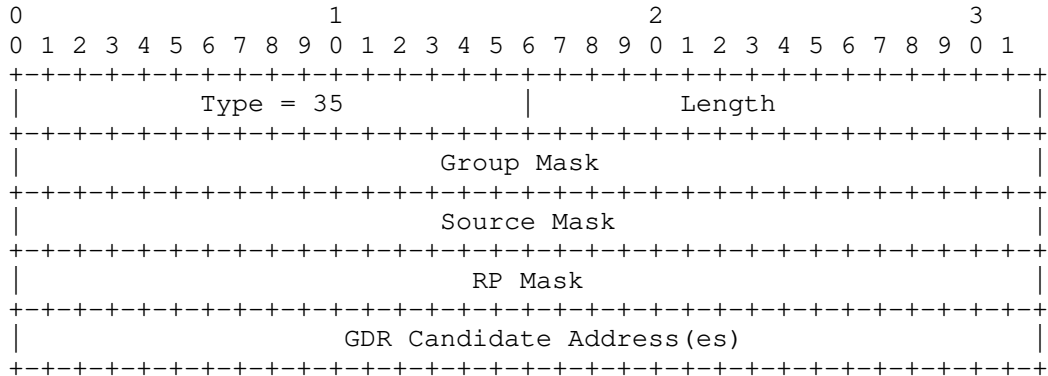


Figure 4: PIM DR Load Balancing List Hello Option

Type: 35

Length: $(3 + n) \times (4 \text{ or } 16)$ bytes, where n is the number of GDR candidates.

Group Mask (32/128 bits): Mask applied to group addresses as part of hash computation.

Source Mask (32/128 bits): Mask applied to source addresses as part of hash computation.

RP Mask (32/128 bits): Mask applied to RP addresses as part of hash computation.

All masks MUST have the same number of bits as the IP source address in the PIM Hello IP header.

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

All addresses MUST be in the same address family as the PIM Hello IP header. It is recommended that the addresses are sorted in descending order.

If the "Interface ID" option, as specified in [RFC6395], is present in a GDR Candidate's PIM Hello message, and the "Router Identifier" portion is non-zero:

- + For IPv4, the "GDR Candidate Address" will be set directly to the "Router Identifier".
- + For IPv6, the "GDR Candidate Address" will be 96 bits of zeroes followed by the 32 bit Router Identifier.

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

This DRLB-List Hello Option MUST only be advertised by the elected PIM DR. It MUST be ignored if received from a non-DR. The option MUST also be ignored if the hash masks are not the correct number of bits, or GDR Candidate addresses are in the wrong address family.

5.4. PIM DR Operation

The DR election process is still the same as defined in [RFC7761]. The DR advertises the new DRLB-List Hello Option, which contains mask values from user configuration (or default values), followed by a list of GDR Candidate Addresses. Note that if a router included the "Interface ID" option in the hello message, and the Router ID is non-zero, the Router ID will be used to form the GDR Candidate address of the router, as discussed in the previous section. It is recommended that the list be sorted, from the highest value to the lowest value. The reason for sorting the list is to make the behavior deterministic, regardless of the order in which the DR learns of new candidates. Note that, as for non-DR routers, the DR also advertises the DRLB-Cap Hello Option to indicate its ability to support the new functionality and the type of GDR election Hash Algorithm it uses.

If a PIM DR receives a neighbor DRLB-Cap Hello Option, which contains the same Hash Algorithm 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' Address into the list of the DRLB-List Option. However, the DR may have policies limiting which GDR Candidates, or the number of GDR Candidates to include. Likewise, the DR SHOULD include itself in the list of GDR Candidates, but it is permissible not to do so, if for instance there is some policy restricting the candidate set.

If a PIM neighbor included in the list expires, stops announcing the DRLB-Cap Hello Option, changes DR priority, changes Hash Algorithm or otherwise becomes ineligible as a candidate, the DR SHOULD

immediately send a triggered hello with a new list in the DRLB-List option, excluding the neighbor.

If a new router becomes eligible as a candidate, there is no urgency in sending out an updated list. An updated list SHOULD be included in the next hello.

5.5. PIM GDR Candidate Operation

When an IGMP/MLD report is received, a Hash Algorithm is used by the GDR Candidates to determine which router is going to be responsible for building forwarding trees on behalf of the host.

The router MUST include the DRLB-Cap Hello Option in all PIM Hello messages sent on the interface. Note that the presence of the DRLB-Cap Option in the PIM Hello does not guarantee that the router will be considered as a GDR candidate. Once the DR election is done, the DRLB-List Hello Option is received from the current PIM DR containing a list of the selected GDRs Candidates.

A router only acts as a GDR Candidate if it is included in the GDR Candidate list of the DRLB-List Hello Option. See next section for details.

5.6. DRLB-List Hello Option Processing

This section discusses processing of the DRLB-List Hello Option, including the case where it was received in the previous hello, but not in the current hello. All routers MUST ignore the DRLB-List Hello Option if it is received from a PIM router which is not the DR. The option MUST only be processed by routers that are announcing the DRLB-Cap Option, and only if the Hash Algorithm announced by the DR is the same as the local announcement. All GDR Candidates MUST use the Hash Masks advertised in the Option, even if they differ from those the candidate was configured with. The DR MUST also process its own DRLB-List Hello Option.

A router stores the latest option contents that was announced, if any, and deletes the previous contents. The router MUST also compare the new contents with any previous contents, and if there are any changes, continue processing as below. Note that if the option does not pass the above checks, the below processing MUST be done as if the option was not announced.

If the contents of the DRLB-List Option, the masks or the candidate list, differs from the previously saved copy, it is received for the first time, or it is no longer being received or accepted, the option MUST be processed as below.

1. If the local router is included in the GDR Candidate Address(es) field (it will look for its own address, or its Router ID if it announces a non-zero Router ID), for each of the groups, or source and group pairs if the group is in SSM mode, with local receiver interest, the router MUST run the Hash Algorithm to determine which of them it is the GDR for.

If there is no change in the GDR status, then no further action is required.

If the router becomes the new GDR, then a multicast forwarding tree MUST be built [RFC7761].

If the router is no longer the GDR, then it uses an Assert as explained in [Section 5.7].

2. If the local router is not included in the GDR Candidate Address(es) field, or if the DRLB-List Hello Option is no longer included in the DR's Hello, or if the DR's Neighbor Liveness Timer expires [RFC7761], for each of the groups, or source and group pairs if the group is in SSM mode, with local receiver interest, for which the router is the GDR, it uses an Assert as explained in [Section 5.7].

5.7. PIM Assert Modification

GDR changes may occur due to configuration change, due to GDR candidates going down, and also new routers coming up and becoming GDR candidates. This may occur while flows are being forwarded. If the GDR for an active flow changes, there is likely to be some disruption, such as packet loss or duplicates. By using asserts, packet loss is minimized, while allowing a small amount of duplicates.

When a router stops acting as the GDR for a group, or source and group pair if SSM, it MUST set the Assert metric preference to maximum (0x7fffffff) and the Assert metric to one less than maximum (0xffffffff). That is, whenever it sends or receives an Assert for the group, it must use these values as the metric preference and metric rather than the values provided by the unicast routing protocol.

The rest of this section is just for illustration purposes and not part of the protocol definition.

To illustrate the behavior when there is a GDR change, consider the following scenario where there are two flows G1 and G2. R1 is the GDR for G1, and R2 is the GDR for G2. When R3 comes up, 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.

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.

For G1, using the functionality described in this document, R1 and R3 determine the new GDR, which is R3. With the modified Assert behavior, R1 sets its Assert metric to the near maximum value discussed above. That will make R3, which has normal metric in its Assert as the Assert winner.

5.8. Backward Compatibility

In the case of a hybrid Ethernet shared LAN (where some PIM routers support the functionality defined in this document, and some do not);

- o If the DR does not support the new functionality, then there will be no load-balancing.
- o If non-DR routers do not support the new functionality, they will not be considered as Candidate GDRs and it will not take part in load-balancing. Load-balancing may still happen on the link.

6. Operational Considerations

An administrator needs to consider what the total bandwidth requirements are and find a set of routers that together has enough available capacity, while making sure that each of the routers can handle its part, assuming that the traffic is distributed roughly equally among the routers. Ideally, one should also have enough bandwidth to handle the case where at least one router fails. All routers should have reachability to the sources, and RPs if applicable, that is not via the LAN.

Care must be taken when choosing what hash masks to configure. One would typically configure the same masks on all the routers, so that they are the same, regardless of which router is elected as DR. The default masks are likely suitable for most deployment. The RP Hash Mask must be configured (the default is no bits set) if one wishes to hash based on the RP address rather than the group address for ASM. The default masks will use the entire group addresses, and source addresses if SSM, as part of the hash. An administrator may set

other masks that masks out part of the addresses to ensure that certain flows always get hashed to the same router. How this is achieved depends on how the group addresses are allocated.

Only the routers announcing the same Hash Algorithm as the DR would be considered as GDR candidates. Network administrators need to make sure that the desired set of routers announce the same algorithm. Migration between different algorithms is not considered in this document.

7. IANA Considerations

IANA has temporarily assigned type 34 for the PIM DR Load Balancing Capability (DRLB-Cap) Hello Option, and type 35 for the PIM DR Load Balancing List (DRLB-List) Hello Option in the PIM-Hello Options registry. IANA is requested to make these assignments permanent when this document is published as an RFC. Note that the option names have changed slightly since the temporary assignments were made. Also, the length of option 34 is always 4, the registry currently says it is variable.

This document requests IANA to create a registry called "Designated Router Load Balancing Hash Algorithms" in the "Protocol Independent Multicast (PIM)" branch of the registry tree. The registry lists Hash Algorithms for use by PIM Designated Router Load Balancing.

7.1. Initial registry

The initial content of the registry should be as follows.

Type	Name	Reference
0	Modulo	This document
1-255	Unassigned	

7.2. Assignment of new Hash Algorithms

Assignment of new Hash Algorithms is done according to the "IETF Review" model, see [RFC8126].

8. Security Considerations

Security of the new DR Load Balancing PIM Hello Options is only guaranteed by the security of PIM Hello messages, so the security

considerations for PIM Hello messages as described in PIM-SM [RFC7761] apply here.

If the DR is subverted it could omit or add certain GDRs or announce an unsupported algorithm. If another router is subverted, it could be made DR and cause similar issues. While these issues are specific to this specification, they are not that different from existing attacks such as subverting a DR and lowering the DR priority, causing a different router to become the DR.

If for any reason, the DR includes a GDR in the announced list which announces a different algorithm from what the DR announces, the GDR is required to ignore the announcement, and there will be no router acting as the DR for the flows that hash to that GDR.

If a GDR is subverted, it could potentially be made to stop forwarding all the traffic it is expected to forward. This is also similar today to if a DR is subverted.

An administrator may be able to achieve the desired load-balancing of known flows, but an attacker may send a single high rate flow which is served by a single GDR, or send multiple flows that are expected to be hashed to the same GDR.

9. Acknowledgement

The authors would like to thank Steve Simlo and Taki Millonis for helping with the original idea; Alia Atlas, Bill Atwood, Joe Clarke, Alissa Cooper, Jake Holland, Bharat Joshi, Anish Kachinthaya, Anvitha Kachinthaya, Benjamin Kaduk, Mirja Kuhlewind, Barry Leiba, Ben Niven-Jenkins, Alvaro Retana, Adam Roach, Michael Scharf, Eric Vyncke and Carl Wallace for reviews and comments; and Toerless Eckert and Rishabh Parekh for helpful conversation on the document.

10. References

10.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6395] Gulrajani, S. and S. Venaas, "An Interface Identifier (ID) Hello Option for PIM", RFC 6395, DOI 10.17487/RFC6395, October 2011, <<https://www.rfc-editor.org/info/rfc6395>>.

- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

10.2. Informative References

- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <<https://www.rfc-editor.org/info/rfc3376>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4541] Christensen, M., Kimball, K., and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, DOI 10.17487/RFC4541, May 2006, <<https://www.rfc-editor.org/info/rfc4541>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.

Authors' Addresses

Yiqun Cai
Alibaba Group

Email: yiqun.cai@alibaba-inc.com

Heidi Ou
Alibaba Group

Email: heidi.ou@alibaba-inc.com

Sri Vallepalli
Cisco Systems, Inc.
3625 Cisco Way
San Jose CA 95134
USA

Email: svallepa@cisco.com

Mankamana Mishra
Cisco Systems, Inc.
821 Alder Drive,
Milpitas CA 95035
USA

Email: mankamis@cisco.com

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

Email: stig@cisco.com

Andy Green
British Telecom
Adastral Park
Ipswich IP5 2RE
United Kingdom

Email: andy.da.green@bt.com

PIM Working Group
Internet-Draft
Intended Status: Standard Track
Expires: April 20, 2018

X. Liu
Jabil
F. Guo
Huawei
M. Sivakumar
Cisco
P. McAllister
Metaswitch Networks
A. Peter
Juniper Networks
Oct 20, 2017

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/lid-abstracts.txt>

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

This Internet-Draft will expire on 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

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

Abstract

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

Table of Contents

1. Introduction	2
1.1. Requirements Language.....	3
1.2. Terminology	3
2. Design of Data model.....	3
2.1. Scope of model	3
2.2. Optional capabilities.....	3
2.3. Position of address family in hierarchy.....	4
3. Module Structure	4
3.1. IGMP Configuration and Operational state.....	4
3.2. MLD Configuration and Operational State.....	6
3.3. IGMP and MLD RPC.....	8
4. IGMP and MLD YANG Modules.....	9
5. Security Considerations.....	32
6. IANA Considerations	32
7. Acknowledgments	33
8. Contributing Authors.....	33
9. References	33
9.1. Normative References.....	33
9.2. Informative References.....	34

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-al
ert}?
      +--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-sourc
es}?
        +--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-su
bnet}?
        +--rw explicit-tracking?             boolean {_intf-explicit-track
ing}?
        +--rw exclude-lite?                 boolean {_intf-exclude-lite}?
        +--rw version?                       uint8
        +--rw join-group*                    inet:ipv4-address {_intf-join
-group}?
        +--rw ssm-map* [source-addr group-policy] {_intf-ssm-map}?
        |   +--rw source-addr                ssm-map-ipv4-addr-type
        |   +--rw group-policy               string
        +--rw static-group* [group-addr source-addr] {_intf-static-group}
?
        |   +--rw group-addr                 inet:ipv4-address
        |   +--rw source-addr                source-ipv4-addr-type
        +--ro oper-status?                   enumeration
        +--ro querier?                       inet:ipv4-address
        +--ro joined-group*                  inet:ipv4-address {_intf-join
-group}?
        +--ro group* [group-address]
        |   +--ro group-address              inet:ipv4-address
        |   +--ro expire?                    uint32
        |   +--ro filter-mode?               enumeration
        |   +--ro up-time?                   uint32
        |   +--ro last-reporter?             inet:ipv4-address
        +--ro source* [source-address]
        |   +--ro source-address             inet:ipv4-address
        |   +--ro expire?                    uint32
        |   +--ro up-time?                   uint32
        |   +--ro host-count?                uint32 {_intf-explicit-tracking}?
        |   +--ro last-reporter?             inet:ipv4-address
        +--ro host* [host-address] {_intf-explicit-tracking}?
        |   +--ro host-address               inet:ipv4-address
        |   +--ro host-filter-mode?          enumeration

```

3.2. MLD Configuration and Operational State

The MLD 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-al
ert}?
+--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-sourc
es}?
    +--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-su
bnet}?
    +--rw explicit-tracking?          boolean {_intf-explicit-track
ing}?
    +--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]
  +--ro group-address      inet:ipv6-address
  +--ro expire?            uint32
  +--ro filter-mode?       enumeration
  +--ro up-time?           uint32
  +--ro last-reporter?     inet:ipv6-address
  +--ro source* [source-address]
    +--ro source-address   inet:ipv6-address
    +--ro expire?          uint32
    +--ro up-time?         uint32
    +--ro host-count?      uint32 {_intf-explicit-tracking}?
    +--ro last-reporter?   inet:ipv6-address
    +--ro host* [host-address] {_intf-explicit-tracking}?
      +--ro host-address    inet:ipv6-address
      +--ro host-filter-mode? enumeration

```

3.3. IGMP and MLD 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-mlld-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>
```

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

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

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

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

revision 2017-10-20 {
  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.";
    }
  }
}
```

```
    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";
        }
    }
}
```

```
    }
    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 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";
    }
  }
}
```

```
    }
    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;
    }
  }
}
```

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

grouping interface-state-attributes-igmp {

    description
        "Per interface state attributes for IGMP.";

    uses interface-state-attributes-igmp-mlld;

    leaf querier {
        type inet:ipv4-address;
        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-mlld;

    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-mlld;
        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
```

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

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

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

grouping interface-state-attributes-mld {

  description
    "Per interface state attributes for MLD.";

  uses interface-state-attributes-igmp-mld;

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

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

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

leaf group-address {
    type inet:ipv6-address;
    description
        "Multicast group address.";
}
uses interface-state-group-attributes-igmp-mlld;

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-mlld;
leaf last-reporter {
    type inet:ipv6-address;
    description
        "The last host address which has sent the
        report to join the multicast source and group.";
}
list host {
    if-feature intf-explicit-tracking;
    key "host-address";
    description
        "List of multicast membership hosts
        of the specific multicast source-group.";

    leaf host-address {
        type inet:ipv6-address;
        description
            "Multicast membership host address.";
    }
    leaf host-filter-mode {
        type enumeration {
            enum "include" {
```



```
        description
            "In include mode";
    }
    enum "exclude" {
        description
            "In exclude mode.";
    }
}
description
    "Filter mode for a multicast membership
    host may be either include or exclude.";
}
} // list host
} // list source
} // list group
} // interface-state-attributes-mlld

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

    leaf expire {
        type uint32;
        units seconds;
        description
            "The time left before multicast group state expires.";
    }
    leaf filter-mode {
        type enumeration {
            enum "include" {
                description
                    "In include mode, reception of packets sent
                    to the specified multicast address is requested
                    only from those IP source addresses listed in the
                    source-list parameter";
            }
            enum "exclude" {
                description
                    "In exclude mode, reception of packets sent
                    to the given multicast address is requested
                    from all IP source addresses except those
                    listed in the source-list parameter.";
            }
        }
    }
    description
        "Filter mode for a multicast group,
        may be either include or exclude.";
}
```

```
    leaf up-time {
        type uint32;
        units seconds;
        description
            "The elapsed time since the device created multicast group record.";
    }
} // interface-state-group-attributes-igmp-mld

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

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

/*
 * Configuration and Operational state data nodes (NMDA version)
 */
augment "/rt:routing/rt:control-plane-protocols/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
```

```

        "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/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

```

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

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

    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
```

```
/*
 * Notifications
 */
}
<CODE ENDS>
```

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]:

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

Registrant Contact: The IESG.

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

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

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

7. 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

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010
- [RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, January 2011
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016
- [RFC8022] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", RFC 8022, November 2016
- [I-D.dsdt-nmda-guidelines] M. Bjorklund, J. Schoenwaelder, P. Shafer, K. Watsen, R. Wilton, "Guidelines for YANG Module Authors (NMDA)", draft-dsdt-nmda-guidelines-01, May 2017
- [draft-bjorklund-netmod-rfc7223bis-00] M. Bjorklund, "A YANG Data Model for Interface Management", draft-bjorklund-netmod-rfc7223bis-00, August 2017

- [draft-bjorklund-netmod-rfc7277bis-00] M. Bjorklund, "A YANG Data Model for IP Management", draft-bjorklund-netmod-rfc7277bis-00, August 2017
- [I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-14, September 2017
- [I-D.acee-netmod-rfc8022bis] L. Lhotka, A. Lindem and Y.Qu, "A YANG Data Model for Routing Management (NDMA Version)", draft-acee-netmod-rfc8022bis-02, September 2017

9.2. Informative References

- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.
- [RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
- [RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [RFC4541] M. Christensen, K. Kimball and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, May 2006.
- [RFC4605] B. Fenner, H. He, B. Haberman, and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.
- [RFC5790] H. Liu, W. Cao and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, February 2010

Authors' Addresses

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

EMail: Xufeng_Liu@jabil.com

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

Email: guofeng@huawei.com

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

Email: masivaku@cisco.com

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

EMail: pete.mcallister@metaswitch.com

Anish Peter
Individual

EMail: anish.ietf@gmail.com

PIM Working Group
Internet-Draft
Intended Status: Standard Track
Expires: December 14, 2019

X. Liu
Volta Networks
F. Guo
Huawei
M. Sivakumar
Juniper
P. McAllister
Metaswitch Networks
A. Peter
Individual
June 14, 2019

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

Status of this Memo

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

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

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

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

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

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

Copyright Notice

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

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

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

Abstract

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

Conventions used in this document

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

Table of Contents

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

1. Introduction

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

This document defines a YANG data model that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices. The protocol versions include IGMPv1 [RFC1112], IGMPv2 [RFC2236], IGMPv3 [RFC3376], MLDv1 [RFC2710], and MLDv2 [RFC3810]. The core features of the IGMP and MLD protocols are defined as required. Non-core features are defined as optional in the provided data model.

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

1.1. Terminology

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

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

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

IGMP:

Internet Group Management Protocol [RFC3376].

MLD:

Multicast Listener Discovery [RFC3810].

SSM:

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

1.2. Tree Diagrams

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

1.3. Prefixes in Data Node Names

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

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

Table 1: Prefixes and Corresponding YANG Modules

2. Design of Data model

2.1. Scope of Model

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

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

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

The model contains all the basic configuration parameters to operate the protocols listed above. Depending on the implementation choices, some systems may not allow some of the advanced parameters to be

configurable. The occasionally implemented parameters are modeled as optional features in this model, while the rarely implemented parameters are not included in this model and left for augmentation. This model can be extended, and has been structured in a way that such extensions can be conveniently made.

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

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

2.1.1. Parameters Not Covered at Global Level

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

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

2.1.2. Parameters Not Covered at Interface Level

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

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

- o Query before immediate leave
- o Send router alert

2.2. Optional Capabilities

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

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

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

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

2.3. Position of Address Family in Hierarchy

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

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

- o The model can support IGMP (IPv4), MLD (IPv6), or both optionally and independently. Such flexibility cannot be achieved cleanly with a combined branch.
- o The structure is consistent with other YANG models such as RFC 8344, which uses separate branches for IPv4 and IPv6.

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

3. Module Structure

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

```

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

```

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

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

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

3.1. IGMP Configuration and Operational State

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

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

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

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

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

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

```

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

```

```

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

```

3.2. MLD Configuration and Operational State

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

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

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

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

Interface level: Including MLD configuration and operational state attributes specific to the given interface. For a configuration node at the interface level, there may exist a

corresponding configuration node with the same name at the interface-global level. The value configured on a node at the interface level overrides the value configured on the corresponding node at the interface-global level.

```

augment /rt:routing/rt:control-plane-protocols
  /rt:control-plane-protocol:
    +--rw mld {feature-mld}?
      +--rw global
        +--rw enable?          boolean {global-admin-enable}?
        +--rw max-entries?     uint32 {global-max-entries}?
        +--rw max-groups?      uint32 {global-max-groups}?
        +--ro entries-count?   uint32
        +--ro groups-count?    uint32
        +--ro statistics
          +--ro discontinuity-time?  yang:date-and-time
          +--ro error
            +--ro total?          yang:counter64
            +--ro query?          yang:counter64
            +--ro report?         yang:counter64
            +--ro leave?          yang:counter64
            +--ro checksum?       yang:counter64
            +--ro too-short?      yang:counter64
          +--ro received
            +--ro total?          yang:counter64
            +--ro query?          yang:counter64
            +--ro report?         yang:counter64
            +--ro leave?          yang:counter64
          +--ro sent
            +--ro total?          yang:counter64
            +--ro query?          yang:counter64
            +--ro report?         yang:counter64
            +--ro leave?          yang:counter64
      +--rw interfaces
        +--rw last-member-query-interval?  uint16
        +--rw query-interval?              uint16
        +--rw query-max-response-time?     uint16
        +--rw require-router-alert?        boolean
        |   {intf-require-router-alert}?
        +--rw robustness-variable?          uint8
        +--rw version?                      uint8
        +--rw max-groups-per-interface?     uint32
        |   {intf-max-groups}?
        +--rw interface* [interface-name]
          +--rw interface-name              if:interface-ref
          +--rw last-member-query-interval? uint16
          +--rw query-interval?             uint16
          +--rw query-max-response-time?    uint16
          +--rw require-router-alert?       boolean

```

```

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

```

```

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

```

3.3. IGMP and MLD Actions

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

```

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

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

```

4. IGMP and MLD YANG Module

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

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

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

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

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

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

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

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

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

  organization
    "IETF PIM Working Group";
```


contact

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

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

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

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

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

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

description

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

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

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

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

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

/*
 * Features
 */
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

feature action-clear-groups {
  description
    "Support actions to clear groups.";
```

```
    }

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

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

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

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

```
    }

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

        leaf enable {
            if-feature global-admin-enable;
            type boolean;
            default true;
            description
                "When this grouping is used for IGMP, this leaf indicates
                whether IGMP is enabled ('true') or disabled ('false')
                in the routing instance.
                When this grouping is used for MLD, this leaf indicates
                whether MLD is enabled ('true') or disabled ('false')
                in the routing instance.";
        }

        leaf max-entries {
            if-feature global-max-entries;
            type uint32;
            description
                "When this grouping is used for IGMP, this leaf indicates
                the maximum number of entries in the IGMP instance.
                When this grouping is used for MLD, this leaf indicates
                the maximum number of entries in the MLD instance.
                If this leaf is not specified, the number of entries is not
                limited.";
        }

        leaf max-groups {
            if-feature global-max-groups;
            type uint32;
            description
                "When this grouping is used for IGMP, this leaf indicates
                the maximum number of groups in the IGMP instance.
                When this grouping is used for MLD, this leaf indicates
                the maximum number of groups in the MLD instance.
                If this leaf is not specified, the number of groups is not
                limited.";
        }
    }
} // global-config-attributes
```

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

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

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

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

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

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

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

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

  leaf total {
    type yang:counter64;
    description
      "The number of total messages.";
  }
  leaf query {
    type yang:counter64;
    description
      "The number of query messages.";
  }
  leaf report {
    type yang:counter64;
    description
      "The number of report messages.";
  }
  leaf leave {
    type yang:counter64;
  }
}
```

```
        description
            "The number of leave messages.";
    }
} // global-statistics-sent-received

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

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

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

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



```

type uint16 {
    range "1..31744";
}
units seconds;
description
    "The Query Interval is the interval between General Queries
    sent by the Querier. In RFC 3376, the Querier's Query
    Interval (QQI) is represented from the Querier's Query
    Interval Code in query message as follows:
    If QQIC < 128, QQI = QQIC.
    If QQIC >= 128, QQIC represents a floating-point value as
    follows:
        0 1 2 3 4 5 6 7
    +---+---+---+---+---+---+
    |1| exp | mant |
    +---+---+---+---+---+---+
    QQI = (mant | 0x10) << (exp + 3).
    The maximum value of QQI is 31744.
    The default value is 125.
    This leaf may be configured at the interface level or the
    interface-global level, with precedence given to the value
    at the interface level. If the leaf is not configured at
    either level, the default value is used.";
reference "RFC 3376. Sec. 4.1.7, 8.2, 8.14.2.";
}
leaf query-max-response-time {
    type uint16 {
        range "1..1023";
    }
    units seconds;
    description
        "Query maximum response time specifies the maximum time
        allowed before sending a responding report.
        The default value is 10.
        This leaf may be configured at the interface level or the
        interface-global level, with precedence given to the value
        at the interface level. If the leaf is not configured at
        either level, the default value is used.";
    reference "RFC 3376. Sec. 4.1.1, 8.3, 8.14.3.";
}
leaf require-router-alert {
    if-feature intf-require-router-alert;
    type boolean;
    description
        "Protocol packets should contain router alert IP option.
        When this leaf is not configured, the server uses the
        following rules to determine the operational value of this
        leaf:
        if this grouping is used in IGMP schema and the value of the

```

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

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

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

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

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

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

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

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

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

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

```
leaf enable {
  if-feature intf-admin-enable;
  type boolean;
  default true;
  description
    "When this grouping is used for IGMP, this leaf indicates
    whether IGMP is enabled ('true') or disabled ('false')
    on the interface.
    When this grouping is used for MLD, this leaf indicates
    whether MLD is enabled ('true') or disabled ('false')
    on the interface.";
}
leaf group-policy {
  type leafref {
    path "/acl:acls/acl:acl/acl:name";
  }
  description
    "When this grouping is used for IGMP, this leaf specifies
    the name of the access policy used to filter the
    IGMP membership.
    When this grouping is used for MLD, this leaf specifies
    the name of the access policy used to filter the
    MLD membership.
    The value space of this leaf is restricted to the existing
    policy instances defined by the referenced schema RFC 8519.
    As specified by RFC 8519, the length of the name is between
    1 and 64; a device MAY further restrict the length of this
    name; space and special characters are not allowed.
    If this leaf is not specified, no policy is applied, and
    all packets received from this interface are accepted.";
  reference
    "RFC 8519: YANG Data Model for Network Access Control Lists
    (ACLs)";
}
leaf immediate-leave {
  if-feature intf-immediate-leave;
  type empty;
  description
    "When this grouping is used for IGMP, the presence of this
    leaf requests IGMP to perform an immediate leave upon
    receiving an IGMPv2 leave message.
    If the router is IGMP-enabled, it sends an IGMP last member
    query with a last member query response time. However, the
    router does not wait for the response time before it prunes
    the group.
    When this grouping is used for MLD, the presence of this
    leaf requests MLD to perform an immediate leave upon
    receiving an MLDv1 leave message.
    If the router is MLD-enabled, it sends an MLD last member
```

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

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

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

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

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

```
    type ssm-map-ipv4-addr-type;
    description
        "Multicast source IPv4 address.";
}
leaf ssm-map-group-policy {
    type string;
    description
        "Name of the policy used to define ssm-map rules.
        A device can restrict the length
        and value of this name, possibly space and special
        characters are not allowed. ";
}
}
list static-group {
    if-feature intf-static-group;
    key "group-addr source-addr";
    description
        "A static multicast route, (*,G) or (S,G).
        The version of IGMP must be 3 to support (S,G).";

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

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

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

```
    leaf ssm-map-source-addr {
      type ssm-map-ipv6-addr-type;
      description
        "Multicast source IPv6 address.";
    }
    leaf ssm-map-group-policy {
      type string;
      description
        "Name of the policy used to define ssm-map rules.
        A device can restrict the length
        and value of this name, possibly space and special
        characters are not allowed.";
    }
  }
}
list static-group {
  if-feature intf-static-group;
  key "group-addr source-addr";
  description
    "A static multicast route, (*,G) or (S,G).
    The version of MLD must be 2 to support (S,G).";

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

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

  leaf oper-status {
    type enumeration {
      enum up {
        description
          "Ready to pass packets.";
      }
      enum down {
        description
          "The interface does not pass any packets.";
      }
    }
  }
}
```



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  leaf expire {
    type uint32;
    units seconds;
    mandatory true;
    description
      "The time left before multicast group state expires.";
  }
  leaf filter-mode {
    type enumeration {
      enum "include" {
        description
          "In include mode, reception of packets sent
          to the specified multicast address is requested
          only from those IP source addresses listed in the
          source-list parameter";
      }
      enum "exclude" {
        description
          "In exclude mode, reception of packets sent
          to the given multicast address is requested
          from all IP source addresses except those
          listed in the source-list parameter.";
      }
    }
    mandatory true;
    description
      "Filter mode for a multicast group,
      may be either include or exclude.";
  }
  leaf up-time {
    type uint32;
    units seconds;
    mandatory true;
    description
      "The elapsed time since the device created multicast group
      record.";
  }
}
```

```
    }  
  } // interface-state-group-attributes  
  
  grouping interface-state-source-attributes {  
    description  
      "Per interface state attributes for both IGMP and MLD  
      source-group records.";  
  
    leaf expire {  
      type uint32;  
      units seconds;  
      mandatory true;  
      description  
        "The time left before multicast source-group state expires.";  
    }  
    leaf up-time {  
      type uint32;  
      units seconds;  
      mandatory true;  
      description  
        "The elapsed time since the device created multicast  
        source-group record.";  
    }  
    leaf host-count {  
      if-feature intf-explicit-tracking;  
      type uint32;  
      description  
        "The number of host addresses.";  
    }  
  } // interface-state-source-attributes  
  
  grouping interface-state-host-attributes {  
    description  
      "Per interface state attributes for both IGMP and MLD  
      hosts of source-group records.";  
  
    leaf host-filter-mode {  
      type enumeration {  
        enum "include" {  
          description  
            "In include mode";  
        }  
        enum "exclude" {  
          description  
            "In exclude mode.";  
        }  
      }  
      mandatory true;  
      description
```

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

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

    container igmp {
        if-feature feature-igmp;
        description
            "IGMP configuration and operational state data.";

        container global {
            description
                "Global attributes.";

            uses global-config-attributes;
            uses global-state-attributes;
        }

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

            uses interfaces-config-attributes-igmp {
                if-feature interface-global-config;
                refine query-interval {
                    default 125;
                }
                refine query-max-response-time {
                    default 10;
                }
                refine robustness-variable {
                    default 2;
                }
                refine version {
                    default 2;
                }
            }
        }
    }
}
```

```
}
list interface {
  key "interface-name";
  description
    "List of IGMP interfaces.";

  leaf interface-name {
    type if:interface-ref;
    must "/if:interfaces/if:interface[if:name = current()]/"
      + "ip:ipv4" {
      error-message
        "The interface must have IPv4 configured, either "
        + "enabled or disabled.";
    }
    description
      "Reference to an entry in the global interface list.";
  }
  uses interface-config-attributes-igmp {
    if-feature per-interface-config;
    refine last-member-query-interval {
      must "../version != 1 or "
        + "(not(..version) and "
        + "(../..version != 1 or not(../..version)))" {
      error-message
        "IGMPv1 does not support "
        + "last-member-query-interval.";
    }
  }
  refine max-group-sources {
    must "../version = 3 or "
      + "(not(..version) and (../..version = 3))" {
    error-message
      "The version of IGMP must be 3 to support the "
      + "source specific parameters.";
    }
  }
  refine source-policy {
    must "../version = 3 or "
      + "(not(..version) and (../..version = 3))" {
    error-message
      "The version of IGMP must be 3 to support the "
      + "source specific parameters.";
    }
  }
  refine explicit-tracking {
    must "../version = 3 or "
      + "(not(..version) and (../..version = 3))" {
    error-message
      "The version of IGMP must be 3 to support the "
```

```

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

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

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

```



```

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

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

  container mld {
    if-feature feature-mld;
    description
      "MLD configuration and operational state data.";

    container global {
      description

```

```
    "Global attributes.";

    uses global-config-attributes;
    uses global-state-attributes;
}
container interfaces {
    description
        "Containing a list of interfaces.";

    uses interfaces-config-attributes-mld {
        if-feature interface-global-config;
        refine last-member-query-interval {
            default 1;
        }
        refine query-interval {
            default 125;
        }
        refine query-max-response-time {
            default 10;
        }
        refine require-router-alert {
            default true;
        }
        refine robustness-variable {
            default 2;
        }
        refine version {
            default 2;
        }
    }
    list interface {
        key "interface-name";
        description
            "List of MLD interfaces.";

        leaf interface-name {
            type if:interface-ref;
            must "/if:interfaces/if:interface[if:name = current()]/"
                + "ip:ipv6" {
                error-message
                    "The interface must have IPv6 configured, either "
                    + "enabled or disabled.";
            }
            description
                "Reference to an entry in the global interface list.";
        }
        uses interface-config-attributes-mld {
            if-feature per-interface-config;
            refine max-group-sources {
```

```

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

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

```

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

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

5. Security Considerations

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

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

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

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

igmp-mld:global

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

igmp-mld:interfaces

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

igmp-mld:interfaces/interface

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

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

```
igmp-mld:global
```

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

```
igmp-mld:interfaces
```

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

```
igmp-mld:interfaces/interface
```

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

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

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

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

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

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

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

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

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

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

6. IANA Considerations

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

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

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

Registrant Contact: The IESG.

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

This document registers the following YANG modules in the YANG Module Names registry [RFC6020]:

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

7. Acknowledgments

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

8. Contributing Authors

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

Email: liuyisong@huawei.com

9. References

9.1. Normative References

- [RFC1112] Deering, S., "Host extensions for IP multicasting", STD 5, RFC 1112, August 1989.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
- [RFC3688] Mealling, M., "The IETF XML Registry", RFC 3688, January 2004.
- [RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011.
- [RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, June 2011.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, July 2013.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, August 2016.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, January 2017.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, December 2017.
- [RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, March 2018.
- [RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, March 2018.
- [RFC8343] Bjorklund, M., "A YANG Data Model for Interface Management", RFC 8343, March 2018.
- [RFC8344] M. Bjorklund, "A YANG Data Model for IP Management", RFC8344, March 2018.
- [RFC8349] Lhotka, L., Lindem, A., and Y. Qu, "A YANG Data Model for Routing Management (NMDA Version)", RFC 8349, March 2018.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, August 2018.
- [RFC8519] M. Jethanandani, S. Agarwal, L. Huang and D. Blair, "YANG Data Model for Network Access Control Lists (ACLs)", RFC 8519, March 2019.

9.2. Informative References

- [RFC3569] Bhattacharyya, S., Ed., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, July 2003.
- [RFC4541] M. Christensen, K. Kimball and F. Solensky, "Considerations for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) Snooping Switches", RFC 4541, May 2006.
- [RFC4605] B. Fenner, H. He, B. Haberman, and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.
- [RFC5790] H. Liu, W. Cao and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, February 2010.
- [RFC6636] H. Asaeda, H. Liu and Q. Wu, "Tuning the Behavior of the Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) for Routers in Mobile and Wireless Networks", RFC 6636, May 2012.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, March 2018.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", RFC 8407, October 2018.
- [I-D.ietf-netconf-subscribed-notifications]
Voit, E., Clemm, A., Prieto, A., Nilsen-Nygaard, E., and A. Tripathy, "Customized Subscriptions to a Publisher's Event Streams", draft-ietf-netconf-subscribed-notifications-26 (work in progress), May 2019.
- [I-D.ietf-netconf-yang-push]
Clemm, A., Voit, E., Prieto, A., Tripathy, A., Nilsen-Nygaard, E., Bierman, A., and B. Lengyel, "YANG Datastore Subscription", draft-ietf-netconf-yang-push-25 (work in progress), May 2019.

Authors' Addresses

Xufeng Liu
Volta Networks

Email: xufeng.liu.ietf@gmail.com

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

Email: guofeng@huawei.com

Mahesh Sivakumar
Juniper Networks
1133 Innovation Way
Sunnyvale, California
USA

Email: sivakumar.mahesh@gmail.com

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

Email: pete.mcallister@metaswitch.com

Anish Peter
Individual

Email: anish.ietf@gmail.com

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 30, 2018

A. Gupta
Avi Networks
S. Venaas
Cisco Systems
October 27, 2017

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

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

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

Table of Contents

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

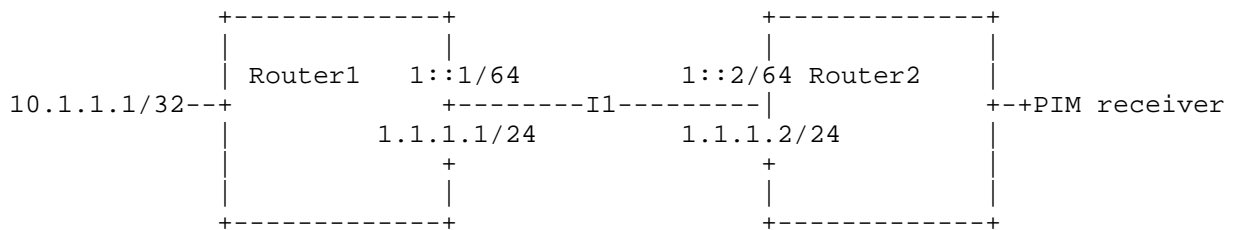
1. Introduction

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 require any additional config changes in the network.

3. Security Considerations

There are no new security considerations.

4. IANA Considerations

There are no IANA considerations.

5. References

5.1. Normative References

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

5.2. Informative References

- [RFC5549] Le Faucheur, F. and E. Rosen, "Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop", RFC 5549, DOI 10.17487/RFC5549, May 2009, <<https://www.rfc-editor.org/info/rfc5549>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: November 2, 2018

A. Gupta
Avi Networks
S. Venaas
Cisco Systems
May 1, 2018

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

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

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

Table of Contents

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

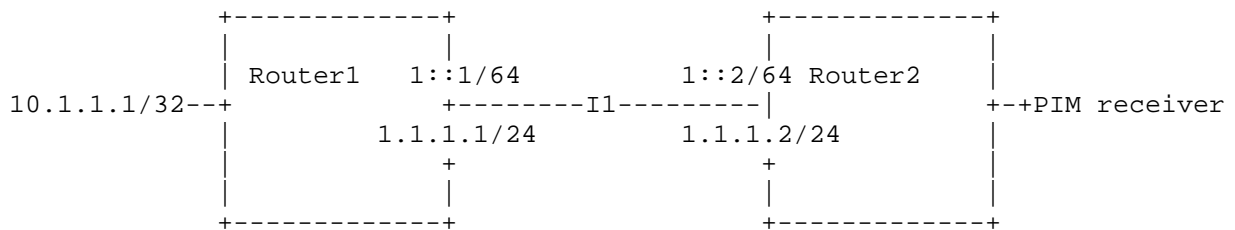
1. Introduction

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 require any additional config changes in the network.

3. Security Considerations

There are no new security considerations.

4. IANA Considerations

There are no IANA considerations.

5. References

5.1. Normative References

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

5.2. Informative References

- [RFC5549] Le Faucheur, F. and E. Rosen, "Advertising IPv4 Network Layer Reachability Information with an IPv6 Next Hop", RFC 5549, DOI 10.17487/RFC5549, May 2009, <<https://www.rfc-editor.org/info/rfc5549>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.

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 WG
Internet-Draft
Intended status: Standards Track
Expires: April 3, 2018

Zheng. Zhang
BenChong. Xu
ZTE Corporation
September 30, 2017

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.

Table of Contents

1. Problem Statement	2
2. Solution	3
2.1. Scheduled Update	4
2.2. Triggered Update	4
3. Message Format	4
3.1. PFM message	4
3.2. BIER information TLV	5
3.3. BIER MPLS Encapsulation sub-TLV	6
3.4. Optional BIER sub-domain BSL conversion sub-TLV	7
4. Security Considerations	7
5. IANA Considerations	7
6. Normative References	7
Authors' Addresses	8

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.

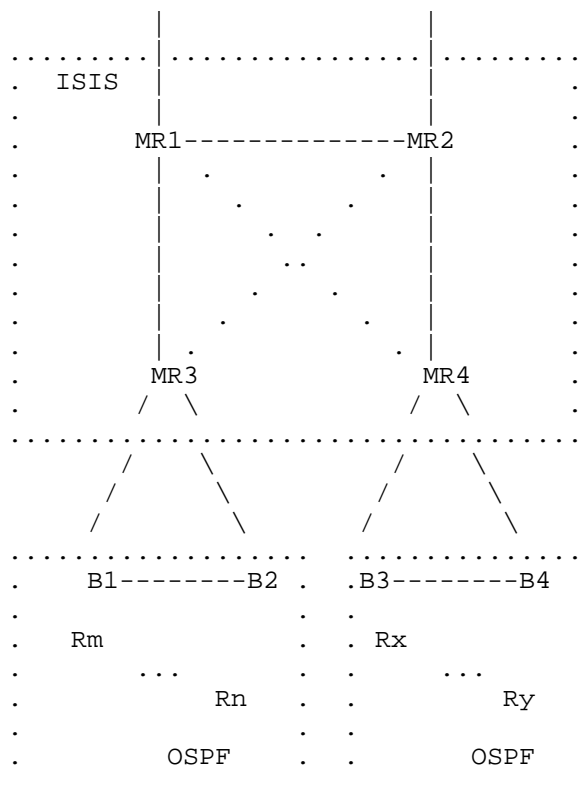


Figure 1: An typical hybrid network

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].

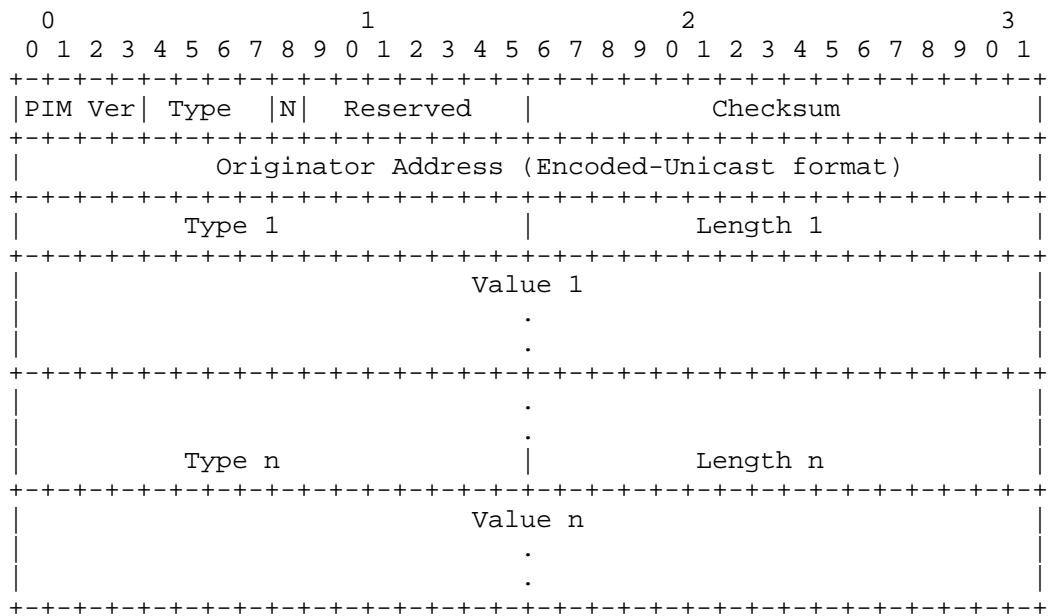


Figure 2: PFM message format

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.

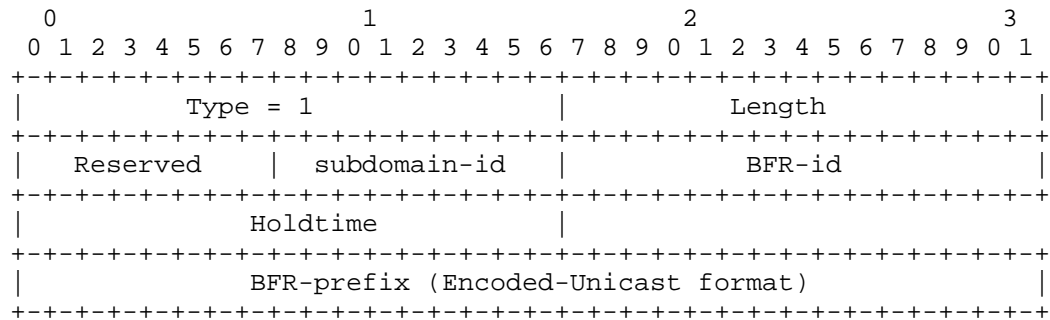


Figure 3: BIER information TLV

- 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 PFM 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]
Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-08 (work in progress), September 2017.
- [I-D.ietf-bier-isis-extensions]
Ginsberg, L., Przygienda, T., Aldrin, S., and Z. Zhang, "BIER support via ISIS", draft-ietf-bier-isis-extensions-05 (work in progress), July 2017.
- [I-D.ietf-bier-ospf-bier-extensions]
Psenak, P., Kumar, N., Wijnands, I., Dolganow, A., Przygienda, T., Zhang, Z., and S. Aldrin, "OSPF Extensions for BIER", draft-ietf-bier-ospf-bier-extensions-07 (work in progress), July 2017.
- [I-D.ietf-pim-source-discovery-bsr]
Wijnands, I., Venaas, S., Brig, M., and A. Jonasson, "PIM flooding mechanism and source discovery", draft-ietf-pim-source-discovery-bsr-06 (work in progress), March 2017.

- [RFC5059] Bhaskar, N., Gall, A., Lingard, J., and S. Venaas, "Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)", RFC 5059, DOI 10.17487/RFC5059, January 2008, <<https://www.rfc-editor.org/info/rfc5059>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.

Authors' Addresses

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

Email: zhang.zheng@zte.com.cn

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

Email: xu.benchong@zte.com.cn

Internet Engineering Task Force
Internet Draft
Intended status: Standards Track
Expires: April 2018

H. Zhao
Ericsson
X. Liu
Jabil
Y. Liu
Huawei
M. Sivakumar
Cisco
A. Peter
Individual

October 26, 2017

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/lid-abstracts.txt>

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

This Internet-Draft will expire on April 26, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction.....	2
1.1. Terminology.....	3
1.2. Tree Diagrams.....	3
2. Design of Data Model.....	3
2.1. Overview.....	4
2.2. IGMP and MLD Snooping Instances.....	4
2.3. IGMP and MLD Snooping References.....	10
2.4. IGMP and MLD Snooping RPC.....	13
3. IGMP and MLD Snooping YANG Module.....	13
4. Security Considerations.....	42
5. IANA Considerations.....	42
6. Normative References.....	42

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:

- o Brackets "[" and "]" enclose list keys.
- o Abbreviations before data node names: "rw" means configuration (read-write), and "ro" means state data (read-only).
- o Symbols after data node names: "?" means an optional node, "!" means a presence container, and "*" denotes a list and leaf-list.
- o Parentheses enclose choice and case nodes, and case nodes are also marked with a colon (":").
- o Ellipsis ("...") stands for contents of subtrees that are not shown.

2. Design of Data Model

The model covers Considerations for Internet Group Management Protocol (IGMP) 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-ml-d-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-al
ert}?
    |      +---rw robustness-variable?                     uint8
    |      +---rw version?                                 uint8
    |      +---rw static-bridge-mrouter-interface*         if:interface-ref {static-l
2-
    multicast-group}?
    |      +---rw static-vpls-mrouter-interface*           l2vpn-instance-pw-ref {sta
tic-
    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

```

```

|   | +--ro mac-address?      yang:phys-address
|   | +--ro expire?          uint32
|   | +--ro up-time?         uint32
|   | +--ro last-reporter?   inet:ipv4-address
|   | +--ro source* [address]
|   |   +--ro address          inet:ipv4-address
|   |   +--ro bridge-outgoing-interface* if:interface-ref
|   |   +--ro vpls-outgoing-ac*  l2vpn-instance-ac-ref
|   |   +--ro vpls-outgoing-pw*  l2vpn-instance-pw-ref
|   |   +--ro up-time?         uint32
|   |   +--ro expire?         uint32
|   |   +--ro host-count?      uint32 {explicit-tracking}
?
|   |   +--ro last-reporter?    inet:ipv4-address
|   |   +--ro host* [host-address] {explicit-tracking}?
|   |     +--ro host-address    inet:ipv4-address
|   |     +--ro host-filter-mode? enumeration
|   +--ro statistics
|     +--ro received
|       | +--ro query?          yang:counter64
|       | +--ro membership-report-v1? yang:counter64
|       | +--ro membership-report-v2? yang:counter64
|       | +--ro membership-report-v3? yang:counter64
|       | +--ro leave?          yang:counter64
|       | +--ro pim?            yang:counter64

```

```

|      +--ro sent
|
|      +---ro query?                yang:counter64
|
|      +---ro membership-report-v1? yang:counter64
|
|      +---ro membership-report-v2? yang:counter64
|
|      +---ro membership-report-v3? yang:counter64
|
|      +---ro leave?                yang:counter64
|
|      +---ro pim?                  yang:counter64
+---rw mld-snooping-instances
|  +---rw mld-snooping-instance* [name]
|
|      +---rw name                    string
|
|      +---rw id?                    uint32
|
|      +---rw type?                  enumeration
|
|      +---rw enable?                boolean {admin-enable}?
|
|      +---rw forwarding-mode?       enumeration
|
|      +---rw explicit-tracking?     boolean {explicit-tracking
}?
|
|      +---rw exclude-lite?          boolean {exclude-lite}?
|
|      +---rw send-query?            boolean
|
|      +---rw fast-leave?            empty {fast-leave}?
|
|      +---rw last-member-query-interval? uint16
|
|      +---rw query-interval?        uint16
|
|      +---rw query-max-response-time? uint16
|
|      +---rw require-router-alert?  boolean {require-router-al
ert}?
|
|      +---rw robustness-variable?   uint8
|
|      +---rw version?               uint8

```

```

2-   |      +---rw static-bridge-mrouter-interface*   if:interface-ref {static-l
    multicast-group}?

    |      +---rw static-vpls-mrouter-interface*      l2vpn-instance-pw-ref {sta
tic-  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

```

```

    |   |   +--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
```

```

ef |                                     +-rw mld-snooping-instance?   mld-snooping-instance-r
+-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
| | +--rw igmp-snooping-instance? igmp-snooping-instan
ce-ref
| | +--rw mld-snooping-instance? mld-snooping-instan
e-ref
| +--:(primary-pw)
|   +--rw primary-pw* [name]
|     +--rw name string
|     +--rw igmp-snooping-instance? igmp-snooping-instan
ce-ref
|     +--rw mld-snooping-instance? mld-snooping-instan
e-ref
+--rw (backup)?
  +--:(backup-ac)
  | +--rw backup-ac
  |   +--rw name? string
  |   +--rw igmp-snooping-instance? igmp-snooping-instan
ce-ref
  |   +--rw mld-snooping-instance? mld-snooping-instan
e-ref
  +--:(backup-pw)
  |   +--rw backup-pw* [name]
  |     +--rw name string
  |     +--rw igmp-snooping-instance? igmp-snooping-instan
ce-ref
  |     +--rw mld-snooping-instance? mld-snooping-instan
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 {
  namespace "urn:ietf:params:xml:ns:yang: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
```

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

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

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

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

Xufeng Liu
<mailto:Xufeng_Liu@jabil.com>

Yisong Liu
<<mailto:liuyisong@huawei.com>>

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

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

";

description

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

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

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

```
revision 2017-06-28 {  
  description  
    "augment /rt:routing/rt:control-plane-protocols
```

```
    augment /rt:routing-state/rt:control-plane-protocols";
  reference
    "RFC XXXX: A YANG Data Model for IGMP and MLD Snooping";
}

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

/*
 * Features
 */

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

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

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

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

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

feature per-instance-config {
  description
    "Support configuration of each VLAN or VPLS instance or EVPN
instance.";
}
```

```
    feature rpc-clear-groups {
        description
            "Support to clear statistics by RPC for IGMP and MLD
Snooping.";
    }

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

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

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

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

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

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

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

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



```
    "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 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";
        }
    }
```

```
    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-mls-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
```

```
        leave latency of the network.";
        reference "RFC3376. Sec. 8.8.";
    }

    leaf query-interval {

        type uint16;
        units seconds;
        default 125;
        description
            "The Query Interval is the interval between General
Queries
        sent by the Querier.";
        reference "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-ml-d-snooping

  grouping instance-config-attributes-ml-d-snooping {
    description "MLD snooping configuration of each VLAN.";

    uses instance-config-attributes-igmp-ml-d-snooping;

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

    list static-l2-multicast-group {
      if-feature static-l2-multicast-group;
    }
  }
}
```

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

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

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

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

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

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

} // static-l2-multicast-group

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

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

    leaf mac-address {
```

```
        type yang:phys-address;
        description "Destination mac address for L2 multicast
forwarding.";
    }

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

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

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

grouping instance-state-attributes-igmp-snooping {
    description
        "State attributes for IGMP snooping for each VLAN or VPLS
instance or EVPN instance.";

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

    list group {
        key "address";
        config false;

        description "IGMP snooping information";

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

    }

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



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

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

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

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

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

  leaf host-address {
    type inet:ipv4-address;
    description
      "Multicast membership host address.";
  }
  leaf host-filter-mode {
    type enumeration {
      enum "include" {
        description
          "In include mode";
      }
      enum "exclude" {
        description
          "In exclude mode.";
      }
    }
  }
}
```

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

    } // list source
} // list group

// statistics
uses general-state-attributes;

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

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

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

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

    leaf-list bridge-mrouter-interface {

        when "ims:type = 'bridge'";
        type if:interface-ref;
        config false;
        description " mrouter interface in bridge forwarding";

    }

    leaf-list vpls-mrouter-interface {

        when "ims:type = 'vpls'";
        type l2vpn-instance-pw-ref;
        config false;
        description " mrouter interface in vpls forwarding";

    }

}
```

```
    }

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

grouping instance-state-attributes-mld-snooping {
  description
    "State attributes for MLD snooping of each VLAN.";

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

  list group {
    key "address";

    config false;

    description "MLD snooping statistics information";

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

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

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

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

      leaf address {
        type inet:ipv6-address;
        description "Source IP address for multicast stream";
      }

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

      leaf last-reporter {
```

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

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

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

} // list source
} // list group

// statistics
uses general-state-attributes;

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

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

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

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

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

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

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

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

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

grouping general-statistics-error {
  description
```

```
    "A grouping defining statistics attributes for errors.";

    leaf checksum {
        type yang:counter64;
        description
            "The number of checksum errors.";
    }
    leaf too-short {
        type yang:counter64;
        description
            "The number of messages that are too short.";
    }
} // general-statistics-error

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

    leaf query {
        type yang:counter64;
        description
            "The number of query messages.";
    }
    leaf membership-report-v1 {
        type yang:counter64;
        description
            "The number of membership report v1 messages.";
    }
    leaf membership-report-v2 {
        type yang:counter64;
        description
            "The number of membership report v2 messages.";
    }
    leaf membership-report-v3 {
        type yang:counter64;
        description
            "The number of membership report v3 messages.";
    }
    leaf leave {
        type yang:counter64;
        description
            "The number of leave messages.";
    }
    leaf pim {
        type yang:counter64;
        description
            "The number of pim hello messages.";
```

```

    }
  } // 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;

```

```

        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
}
}
}
}

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

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

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

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

```

```
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.";
                    }
                }
            }
            leaf igmp-snooping-instance {
                type igmp-snooping-instance-ref;
                description "Configure igmp-snooping instance
under the vlan view";
            }
            leaf mld-snooping-instance {
                type mld-snooping-instance-ref;
                description "Configure mld-snooping instance
under the vlan view";
            }
        }
        container interfaces {
            description
                "Interface configuration parameters.";
        }

```

```
list interface {
    key "name";

    description
        "The list of configured interfaces on the
device.";

    leaf name {
        type string;
        description
            "The name of the interface.";
    }
    leaf igmp-snooping-instance {
        type igmp-snooping-instance-ref;
        description "Configure igmp-snooping
instance under the interface view";
    }
    leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping
instance under the interface view";
    }
}
} //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
12vpn-instance view";
    }
    leaf mld-snooping-instance {
        type mld-snooping-instance-ref;
        description "Configure mld-snooping instance under the
12vpn-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-mlld-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  
}  
<CODE ENDS>
```

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

- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010.
- [RFC6021] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6021, 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.
- [RFC2236] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.
- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, October 1999.

- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.
- [RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.
- [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using InternetGroup Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, August 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.
- [draft-ietf-pim-igmp-mld-yang-01] X. Liu, F. Guo, M. Sivakumar, P. McAllister, A. Peter, "A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)", draft-ietf-pim-igmp-mld-yang-01, October 28, 2016.
- [draft-ietf-pim-igmp-mld-yang-03] X. Liu, F. Guo, M. Sivakumar, P. McAllister, A. Peter, "A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)", draft-ietf-pim-igmp-mld-yang-03, March 13, 2017.
- [draft-dsdt-nmda-guidelines-01] M. Bjorklund, J. Schoenwaelder, P. Shafer, K. Watsen, R. Wilton, "Guidelines for YANG Module Authors (NMDA)", draft-dsdt-nmda-guidelines-01, May 2017
- [draft-bjorklund-netmod-rfc7223bis-00] M. Bjorklund, "A YANG Data Model for Interface Management", draft-bjorklund-netmod-rfc7223bis-00, August 21, 2017
- [draft-bjorklund-netmod-rfc7277bis-00] M. Bjorklund, "A YANG Data Model for IP Management", draft-bjorklund-netmod-rfc7277bis-00, August 21, 2017
- [draft-ietf-netmod-revised-datastores-03] M. Bjorklund, J. Schoenwaelder, P. Shafer, K. Watsen, R. Wilton, "Network Management Datastore Architecture", draft-ietf-netmod-revised-datastores-03, July 3, 2017
- [draft-ietf-bess-evpn-yang-02] P. Brissette, A. Sajassi, H. Shah, Z. Li, H. Chen, K. Tiruveedhula, I. Hussain, J. Rabadan, "Yang Data Model for EVPN", draft-ietf-bess-evpn-yang-02, March 13, 2017

[draft-ietf-bess-l2vpn-yang-06] H. Shah, P. Brissette, I. Chen, I. Hussain, B. Wen, K. Tiruveedhula, "YANG Data Model for MPLS-based L2VPN", draft-ietf-bess-l2vpn-yang-06.txt, June 30, 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

EMail: masivaku@cisco.com

