

SPRING Working Group
Internet Draft
Intended status: Standards Track
Expires: August 2016

Dave Allan, Jeff Tantsura
Ericsson

February 2016

A Framework for Computed Multicast applied to MPLS based Segment
Routing
draft-allan-spring-mpls-multicast-framework-00

Abstract

This document describes a multicast solution for Segment Routing with MPLS data plane. It is consistent with the Segment Routing architecture in that an IGP is augmented to distribute information in addition to the link state. In this solution it is multicast group membership information sufficient to synchronize state in a given network domain. Computation is employed to determine the topology of any loosely specified multicast distribution tree.

Status of this Memo

This Internet-Draft is submitted to IETF 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 August 2016.

Copyright and License Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction.....	3
1.1. Authors.....	3
1.2. Requirements Language.....	3
2. Conventions used in this document.....	3
2.1. Terminology.....	3
3. Solution Overview.....	4
3.1. Mapping source specific trees onto the segment routing architecture.....	5
3.2. Role of the Routing System.....	5
3.3. MDT Construction Requirements.....	6
3.4. Pruning - theory of operation.....	6
4. Elements of Procedure.....	7
4.1. Triggers for Computation.....	7
4.2. FIB Determination.....	7
4.2.1. Information in the IGP.....	7
4.2.2. Computation of individual segments.....	7
4.3. FIB Generation.....	10
4.4. FIB installation.....	10
5. Related work.....	11
5.1. IGP Extensions.....	11
5.2. BGP Extensions.....	11
6. Observations.....	11
7. Acknowledgements.....	12
8. Security Considerations.....	12
9. IANA Considerations.....	12
10. References.....	12
10.1. Normative References.....	12
10.2. Informative References.....	12
11. Authors' Addresses.....	13

1. Introduction

This memo describes a solution for multicast for Segment Routing with MPLS data plane in which source specific multicast distribution trees (MDTs) are computed from information distributed via an IGP. Computation can use information in the IGP to determine if a given node in the network has a role as a root, leaf or replication point in a given MDT. Unicast tunnels are employed to interconnect the nodes determined to have a role. Therefore state only need be installed in nodes that have one of these three roles to fully instantiate an MDT.

Although this approach is computationally intensive, a significant amount of computation can be avoided when the computing agent determines that the node it is computing for has no role in a given MDT. This permits a computed approach to multicast convergence to be computationally tractable.

1.1. Authors

David Allan, Jeff Tantsura

1.2. 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 RFC2119 [RFC2119].

2. Conventions used in this document

2.1. Terminology

Candidate replication point - is a node that potentially needs to install state to replicate multicast traffic as determined at an intermediate step in multicast segment computation. It will either resolve to having no role or a role as a replication point once multicast has converged.

Candidate role - refers to any potential combination of roles on a given multicast segment as determined at some intermediate step in MDT computation. For example, a node with a candidate role may be a leaf and may be a candidate replication point.

Downstream - refers to the direction along the shortest path to one or more leaves for a given multicast distribution tree

Multicast convergence - is when all computation and state installation to ensure the FIB reflects the multicast information in the IGP is complete.

MDT - multicast distribution tree. Is a tree composed of one or more multicast segments.

Multicast segment - is a portion of the multicast tree where only the root and the leaves have been specified, and computation based upon the current state of the IGP database will be employed to determine and install the required state to implement the segment. A multicast segment is identified by a multicast SID.

Pinned path - Is a unique shortest path extending from a leaf upstream towards the root for a given multicast segment. Therefore is a component of the multicast segment that it has been determined must be there. It will not necessarily extend from the leaf all the way to the root during intermediate computation steps. A pinned path can result from pruning operations.

Role - refers specifically to a node that is either a root, a leaf, a replication node, or a pinned waypoint for a given MDT.

Unicast convergence - is when all computation and state installation to ensure the FIB reflects the unicast information in the IGP is complete.

Upstream - refers to the direction along the shortest path to the root of a given MDT.

3. Solution Overview

This memo describes a multicast architecture in which multicast state is only installed in those nodes that have roles as a root, leaves, and replication points for a given multicast segment. The a-priori established segment routing unicast tunnels are used as interconnect between the nodes that have a role in a given multicast SID.

A loosely specified MDT is composed of a single multicast segment and the routing of the MDT is delegated entirely to computation driven by information in the IGP database.

Explicitly routed MDTs are expressed as a tree of concatenated multicast segments where both the leaves of each segment and the waypoints coupling a given segment to the upstream and/or downstream segment(s) is specified in information flooded in the IGP by the

overall root of the MDT. The segments themselves will be computed as per a loosely specified MDT.

A PE acting as an overall root for a given tree is expected to be configured by management as to where to source multicast traffic from, be it an attachment circuit, interworking function for client technology or other. Similarly a leaf for a given tree is expected to be configured by management as to the disposition of received multicast traffic.

A computed segment is guaranteed to be loop free in a stable system. A concatenation of segments to construct an MDT will similarly be loop free as any collision of segments can be disambiguated in the data plane via the SIDs.

This architecture significantly reduces the amount of state that needs to be installed in the data plane to support multicast. This also means that the impact of many failures in the network on multicast traffic distribution will be recovered by unicast local repair or unicast convergence with subsequent multicast convergence acting in the role of network re-optimization (as opposed to restoration).

3.1. Mapping source specific trees onto the segment routing architecture

A computed source specific tree for a given multicast group corresponds to one or more multicast segments in the SR architecture, each of which is assigned a SID, typically by management configuration of the node that will be the overall root for the source specific tree, which then uses the IGP to advertise this information to the root's peers.

A multicast group is implemented as the set of source specific trees from all nodes that have registered transmit interest to all nodes that have registered receive interest in a multicast group.

3.2. Role of the Routing System

The role of the IGP is to communicate topology information, multicast registrations, unicast to SID bindings, multicast to SID bindings and waypoints in multi-segment MDTs. No changes to topology or unicast to SID bindings advertisement are proposed by this memo.

The multicast registrations/bindings will be in the form of source, group, transmit/receive interest and the SID to use for the source specific multicast tree. Registrations are originated by any node that has send or receive interest in a given multicast group. Nodes

will use the combination of topology and multicast registrations to determine the nodes that have a role in each source specific tree and the SID information to then derive the required FIB state.

The definition of the required IGP TLVs is out of scope of this memo and will be done in relevant IGP drafts.

3.3. MDT Construction Requirements

A multicast segment in an MDT is constructed such that between any pair of nodes that have a role in the segment and are connected by a unicast tunnel, there is not another node on the shortest path between the two with a role in that segment. This ensures that copies of a packet forwarded by an multicast segment will traverse a link only once in a stable system.

Note that this can be satisfied by a minimum cost shortest path tree, but is not an absolute requirement. The pruning rules specified in this memo will meet this requirement without necessarily producing absolutely minimum cost multicast segment (or incurring the associated computational cost).

3.4. Pruning - theory of operation

The role of nodes in a given multicast segment is determined by first producing an inclusive shortest path tree with all possible paths between the root and leaves, and then applying a set of pruning rules repeatedly until an acyclic tree is produced or no further prunes are possible.

For the majority of multicast segments these rules will authoritatively produce a minimum cost tree. For those segments that have not yet been authoritatively resolved, there is a set of pruning operations applied that are not guaranteed to produce a tree that meets the requirements of 3.3, therefore these trees require auditing and potential correction according to a further set of agreed rules. This avoids the necessity of an exhaustive search of the solution space.

A node during computation of a segment may conclude that it will absolutely not have a role at any of numerous points in the computation process and abandon computation of that segment.

4. Elements of Procedure

4.1. Triggers for Computation

MDT computation is triggered by changes to the IGP database. These are in the form of either changes in registered multicast group interest, addition or removal of a multi-segment MDT descriptor, or topology changes.

A change in registered interest for a group will require re-computation of all MDTs that implement the multicast group.

A topology change will require the computation of some number of multicast segments, the actual number will depend on the implementation of tree computation but at a minimum will be all trees for which there is not an optimal shortest path solution as a result of the topology change.

4.2. FIB Determination

4.2.1. Information in the IGP

Group membership information for a multicast segment is obtained from the IGP. This is true for single segment MDTs as well as multi-segment MDTs. Included in the multi-segment MDT specification is the waypoint nodes in MDT and the upstream and downstream SIDs. The specified node is expected to cross connect the SIDs to join the segments together acting in the role of leaf for the upstream segment and root for the downstream segment.

When a waypoint in an MDT descriptor does not exist in the IGP, the assumption is that the node has failed. The response of the other nodes in the system in FIB determination is to add the leaves of the downstream segment to the upstream segment.

4.2.2. Computation of individual segments

FIB generation for a multicast segment is the result of computation, ultimately as applied to all source specific trees in the network. All computing nodes implement a common algorithm for tree generation, as all MUST agree on the solution.

One algorithm is as follows:

All possible shortest paths to the set of leaves for the MDT is determined. Then pruning rules are repeatedly applied until no further prunes are possible.

The philosophy of the application of these rules could be expressed as "simplify as much as possible, and prune that which cannot be". The rules are:

- 1) Eliminate any links and nodes not on a potential shortest path from the root to the leaves for the MDT under consideration.
- 2) Simplify via the replacement of any nodes that do not have a potential role in the MDT with links.

This will be nodes that are not a leaf, a root or a candidate replication point. For example:

Root-----A-----B

B is a leaf. A is not but is in a potential shortest path from root to B. However A will have no role in the MDT that serves B as it provides simple transit therefore is replaced with a direct connection between the root and B.

Root-----B

Note that such pruning also needs to avoid the creation of duplicate links. For example:

```

      /-----A-----\
Root                               B
      \-----C-----/

```

Where A and C have no role, they can be replaced with a single link from Root to B.

- 3) Simplify via the elimination of fewer hop paths

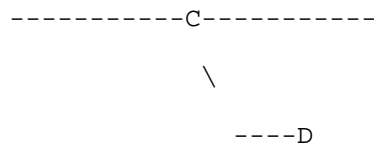
When for a given set of leaves, a node has multiple downstream links that converge on a common downstream point, and that set of leaves is only a subset of the leaves reachable on one or more of the links, any link that only serves that subset of leaves can be pruned.

For example:

```

--A-----B
  \               /

```

B and D are leaves of a root upstream of A. From A, link AB can reach leaf B. Path AC can reach leaf B and D. In this case path A-B can be pruned from consideration. The set of leaves reachable via link A-B is a subset of that reachable by A-C, and the paths from A that serves that subset converges at B.

- 4) Prune via the elimination of upstream links where the nearest reachable leaf is further than the closest leaf or pinned path, and that path does not have a candidate replication point closer than the closet leaf or pinned path, as the resulting tree will require the shortest path to transit the closest upstream leaf or pinned path.

For each upstream link for each leaf in a segment the nearest leaf or pinned path is determined. Those links for which the nearest leaf is further upstream than the closest leaf are pruned.

If, at the end of pruning and simplification, all leaves in a multicast segment have a unique shortest path to the root, the tree is considered resolved, and the computation can progress directly to the FIB generation step.

If not all leaves have a unique shortest path, additional pruning steps are applied. These steps are NOT guaranteed to produce a lowest cost tree, and therefore require an additional audit and possible modification to ensure when forwarding a maximum of one copy of a packet will traverse an interface.

For segments not authoritatively resolved by the above rules, a prune that will not authoritatively result in a minimum cost tree is applied. For the purpose of interoperability, the following rule is proposed: A computing node will select the closest node to the root with a candidate role that does not have a unique shortest path to the root. Where more than one such node exists, the one with the lowest unicast SID is selected. For that node, the best upstream link is selected and all other upstream links pruned. The best upstream link is defined as the link with the closest node with a candidate role that potentially serves the highest number of leaves. Where there is a tie, once again the node with the lowest SID is selected.

Once the links have been pruned, rules 2 through 4 are repeatedly applied until either the tree is fully resolved, or again no further prunes are possible, in which case the next closest remaining unresolved node has the same prune applied.

For all segments not resolved by the initial prune rules, they are audited to ensure all nodes that have a role in the tree do not have a node with a role between them and their upstream node on the tree. If they do, the old upstream adjacency is removed, and the superior one added.

4.3. FIB Generation

The topology components that remain at the end of the pruning operation will reflect all nodes that have a role in a given multicast segment plus the necessary tunnels (as all intervening multi-path scenarios will have been simplified away). From this the FIB can be generated:

All nodes that have a role in a given multicast segment and have nodes upstream in the segment will need to accept the SID for the MDT from at minimum, all upstream interfaces.

All nodes that have a role in a given segment and have nodes immediately downstream in the segment will need to replicate packets simply labelled with the multicast SID onto those interfaces.

All nodes that have a role in a given segment and have nodes reachable via a tunnel downstream set the FIB to push the tunnel unicast SID for the downstream node onto any replicated copies of a received packet, and identify the set of interfaces on the shortest path for the tunnel SID.

4.4. FIB installation

FIB installation needs to acknowledge two aspects of the hybrid tunnel and role model of multicast tree construction. The first is that because of the sparse state model simple tree adds, moves, and changes may require the installation of state where it did not previously exist, and such changes may impact existing services. The second is that it is possible to retain the knowledge to prioritize computation of those trees impacted the failure of a node with a role.

To address this, there are three stages of state installation for multicast convergence:

1) Immediate:

- a. Installation of state for multicast segments impacted by the failure of a node in the network, and installation of state for segments in nodes that have not previously had a role in the given segment.
- b. Installation of state for waypoints in multi-segment MDTs.

2) After T1: Update state for nodes that both had and have a role in a given multicast segment.

3) After T2: Removal of state for nodes that transition from having a role to not having a role for a given multicast segment.

T1 and T2 will be network wide configurable values.

5. Related work

5.1. IGP Extensions

RFC 6329 provides a useful example of some of the type of IGP changes that will be required. There are two aspects in RFC 6329 that are worth emulating:

- The advertisement of multicast registrations
- The negotiation of the algorithm to be used for MDT computation

The required changes for both IS-IS and OSPF will be documented in separate WG targeted I-Ds.

5.2. BGP Extensions

This memo will require the specification of a new PMSI Tunnel Attribute (SPRING P2MP tunnel, tentatively 0x09) to order to integrate into the multicast framework documented in RFC 6514

6. Observations

This technique is not confined to segment routing, and with the provision of a global label space (to be employed as per a multicast SID), an MPLS-LDP network would also provide the requisite mesh of unicast tunnels and be capable of implementing this approach to multicast.

This memo focuses on an implementation based upon nodes that are IGP speakers and converge independently so is written in a form that assumes a node, computing node and IGP speaker are one in the same. It should be observed that the relative frugality of data plane state would suggest that separation of computation from nodes in the data plane combined with management or "software defined networking" based population of the multicast FIB entries may also be useful modes of network operation.

7. Acknowledgements

8. Security Considerations

For a future version of this document.

9. IANA Considerations

For a future version of this document.

10. References

10.1. Normative References

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

10.2. Informative References

[RFC6379] Ashwood-Smith et.al., "IS-IS Extensions Supporting IEEE 802.1aq Shortest Path Bridging", IETF RFC 6329, April 2012

[RFC6514] Aggarwal et.al., "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", IETF RFC 6514, February 2012

[RFC7385] Andersson & Swallow "IANA Registry for P-Multicast Service Interface (PMSI) Tunnel Type Code Points", IETF RFC 7385, October 2014

11. Authors' Addresses

Dave Allan (editor)
Ericsson
300 Holger Way
San Jose, CA 95134
USA
Email: david.i.allan@ericsson.com

Jeff Tantsura
Ericsson
200 Holger Way
San Jose, CA 95134
Email: jeff.tantsura@ericsson.com

PIM Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2016

Hermin Anggawijaya
Allied Telesis Labs, NZ
February 23, 2016

Improving IGMPv3 and MLDv2 Resilience
draft-anggawijaya-pim-resilient-gmp-01

Abstract

Host devices send IGMP or MLD group membership report messages to tell the designated router (DR) which multicast groups they want to receive.

These messages are sent repeatedly up to maximum number of times determined by the 'Robustness Variable' setting. However, in certain situations, those messages could get lost.

This draft proposes a mechanism whereby host devices attached to a local area network where the querier and the DR are the same device, can request the querier to acknowledge each report message it receives, ensuring report messages reception by the DR.

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/lid-abstracts.html>

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

This Internet-Draft will expire in August 26, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.
This document is subject to BCP 78 and the IETF Trust's Legal

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

Table of Contents

1. Introduction	3
2. Terminology	5
3. Resilient GMP	5
3.1 Resilient IGMP	7
3.1.1 Message Formats	7
3.1.2 Host Behaviour	9
3.1.3 Router Behaviour	10
3.1.4 Backward Compatibility	11
3.2 Resilient MLD	12
3.2.1 Message Formats	12
3.2.2 Host Behaviour	15
3.2.3 Router Behaviour	16
3.2.4 Backward Compatibility	17
3.3 Operational Consideration	17
4. IANA Considerations	17
4.1 IGMP Type Numbers	17
4.2 ICMPv6 Type Numbers	17
5. Security Considerations	18
5.1. On-link Query Message Forgery	18
5.2. On-link Membership Report Message Forgery	18
6. Acknowledgements	18
7. References	19
7.1. Normative References	19
7.2. Informative References	19
Authors' Addresses	19

1. Introduction

IGMP and MLD, known collectively as Group Membership Protocols (GMPs) are used between hosts (multicast listeners) and their designated router. Hosts use these messages to tell the DR which multicast groups data they wish to receive.

When a host wants to receive multicast data for a particular multicast group, it sends a membership report message to the DR. Upon receiving this message, the DR translate the message into a multicast join toward the upstream multicast router using multicast routing protocols, for the indicated group and the DR also maintains a state of the group membership.

In order to confirm that hosts still want to continue receiving multicast data, the GMP querier periodically sends query messages to all users of the multicast group. All hosts requiring multicast data are required to send membership response messages to the querier. When the querier receives a membership report for a particular multicast group, the corresponding membership timer is reset. In a LAN with only one multicast router, the DR will be the same device as the querier. In a LAN where multiple multicast routers coexist, although there is no protocol requirements for the querier and the DR to be on the same device, it is quite common for the querier and the DR to be on the same device. In the following discussion, it is assumed that the querier and the DR to be the same device, and the terms querier and DR will be used interchangeably, referring to the same device.

If the querier does not receive a membership report for a particular group before the group membership timer expires, the querier will delete the appropriate membership record, and the DR which maintains the same membership record using the same state machine, sends a multicast leave to its upstream router.

Early physical and link layer technologies such as 802.3 10Base-T, are characterised by the possibility of frames getting lost during transmission. To compensate for the possibility of frame losses, both the hosts and querier are required to repeat their GMP protocol message transmissions. The number of retransmissions is determined by a Robustness Variable (RV) setting, which is announced by the querier. The RV is configured to reflect the robustness of the protocols against the perceived probability of frame loss within the link.

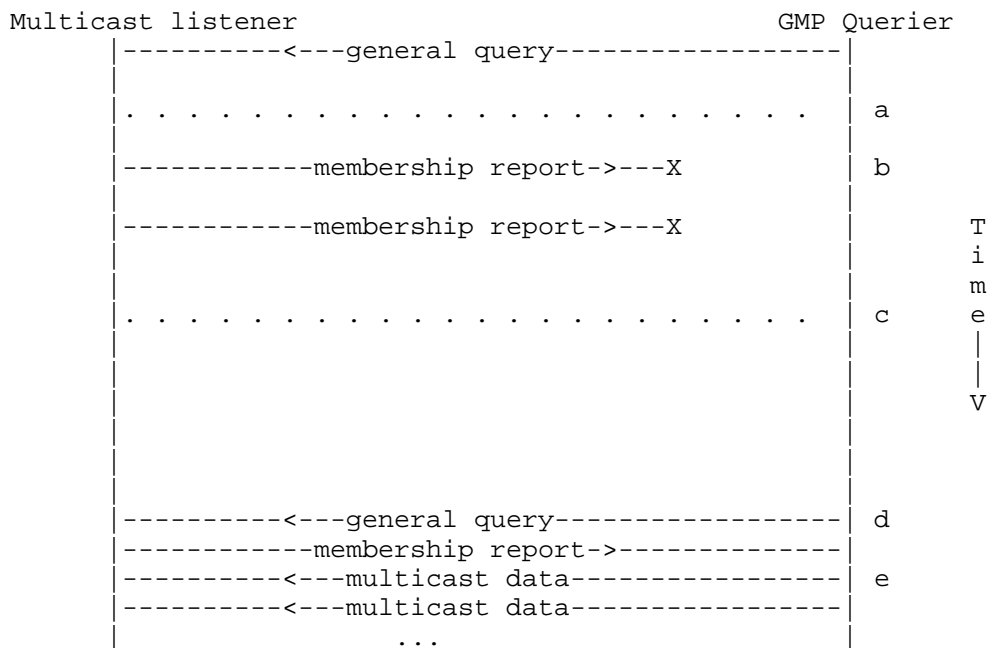
Although wired link layer technology improvements have been made to minimise the possibility of frame loss, the following recent networking trends have tended to negate these effects:

The growing tendency to increase the number of devices operating within a single broadcast domain, then interconnecting multiple domains using either bridged (or layer two switched) links.

For example if a multicast listener sends a group membership report toward a querier located several physical links away, and one of the bridges or L2 switches in the path toward the querier is recovering from a reboot and not in a forwarding state the membership report message sent by the multicast listener could get lost. Multicast packet losses have also been proven to be more prevalent in the ubiquitous wireless (WiFi) link environments. This observation is well documented, see [VYNCKE] and [MCBRIDE].

In the above scenarios, if a membership report message and its subsequent retransmission is lost, then the multicast listener sending the message has to wait until the querier sends a general query message before another message can be sent. With the default GMP protocol values, this waiting period could be up to 125 seconds.

Diagram 1 below illustrates a scenario in which membership report messages are lost between the listener and querier, causing the listener to experience excessive delay in receiving multicast data.



Ta-Tc - transient state period
 Tb-Te - delay for multicast data
 Tc-Td - 'dead' period

Diagram 1. Current GMP operation with transient state on querier

Increasing the RV value may help to alleviate the packet loss issue, but this also make the protocols unnecessarily chatty in normal conditions. This chattiness can have a serious impact if there are lots of multicast listeners in the same broadcast domain.

This draft proposes a new mechanism that allows listeners to send multicast membership reports that are accompanied by a marker signalling the querier to acknowledge the reception of the membership report message in such a way that the multicast listener is sure that its membership report message has been received by the querier. And that the multicast listener does not need to repeat its message transmission RV number of times. However, if the group membership message is lost, the multicast listener can repeat the message transmission until either an acknowledgement is received, a general query is received, or the number of retransmissions reaches the maximum configured on the listener.

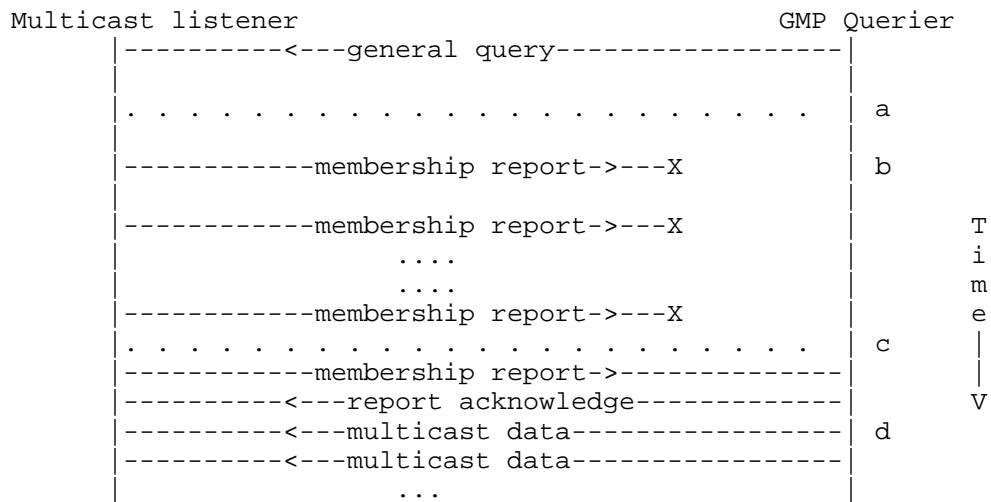
2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [1] and indicate requirement levels for compliant BIDIR-PIM implementations.

3. Resilient GMP

To make GMP more resilient, multicast listeners are allowed to keep sending group membership reports until an acknowledgement of the report is received from the querier. This mechanism is different to the original GMP specifications in that it allows a more robust delivery of membership report messages, and contains less protocol chatter if the link does not suffer from frame losses, i.e. the multicast listener will not unnecessarily retransmit the membership report messages.

Diagram 2 below illustrates the behaviour of multicast listener and querier adhering to the mechanism proposed in this draft.



Ta-Tc - transient state period
Tb-Td - delay for multicast data

Diagram 2. Resilient GMP operation with transient state on querier

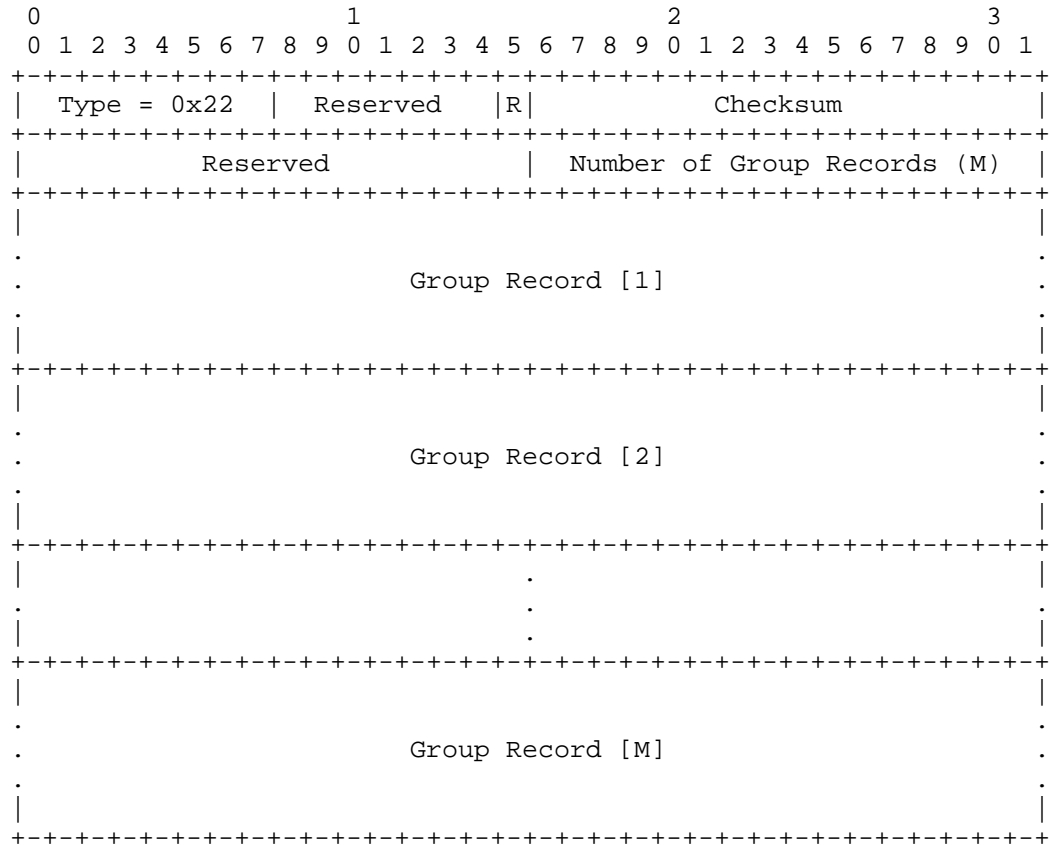
In order to apply this mechanism, a new message format is required that includes a flag to enable multicast listeners to request an acknowledgement from the querier. This message format is defined in the next section.

The new behaviour as specified by this draft is disabled by default and must be enabled explicitly by the network operator.

3.1. Resilient IGMP

3.1.1. Message Formats

IGMPv3 Membership Report

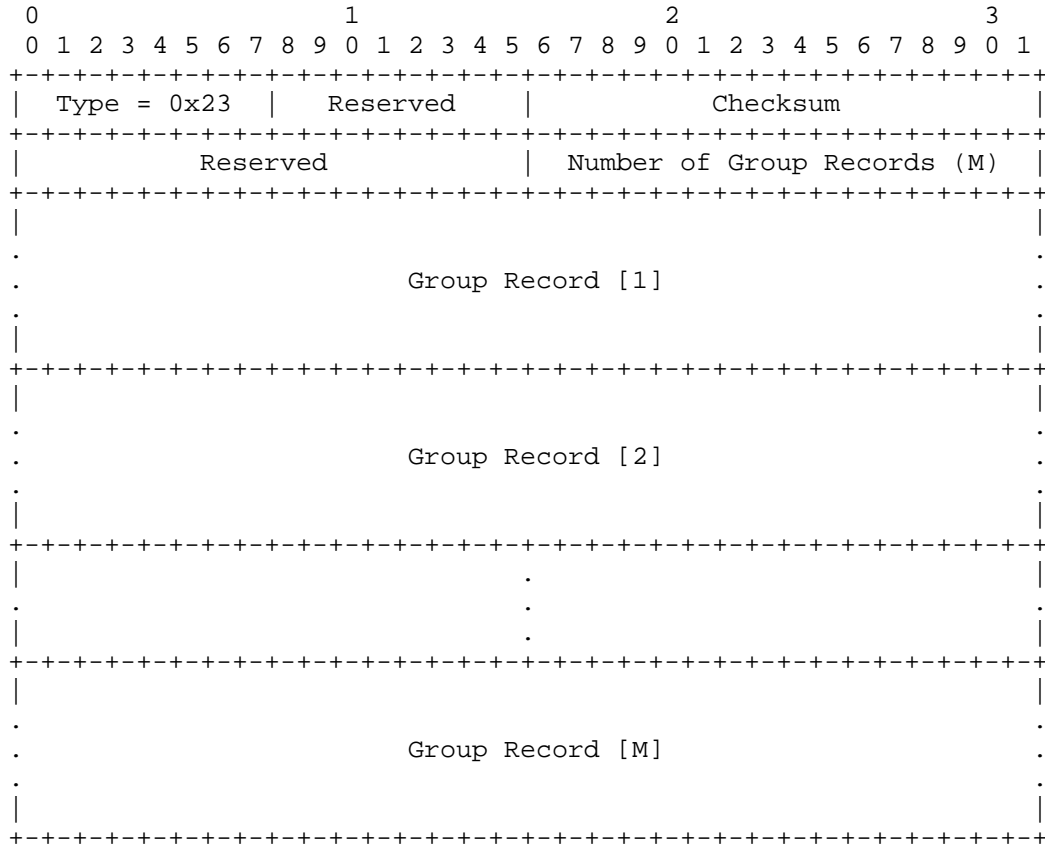


Please see [RFC3376] for the definitions of the existing fields.

New flags defined in the membership report message are:

- R: 'Request for Acknowledgement' flag. Setting this flag indicates that the multicast listener sending the membership report requires an acknowledgement message in reply to the current membership report message.

IGMPv3 Membership Report Acknowledgement



All the fields in the IGMPv3 Membership Report Acknowledgement message have the same specifications as in IGMPv3 Membership Report message, except the type code is 0x23 (to be requested to IANA). The format is chosen so that it is very convenient for the DR to acknowledge membership report messages by simply copying the original membership report message, and modifying the copied packet with new IP source and destination addresses, IGMP packet type, and recalculating both IP and IGMP checksums.

The membership report acknowledgement is sent unicast to the source of the original membership report message being acknowledged.

IGMPv3 Query

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|  Type = 0x11  | Max Resp Code |           Checksum           |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Group Address                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+
| Resv|A|S| QRV |       QQIC       |   Number of Sources (N)   |
+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Source Address [1]                                     |
+-+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Source Address [2]                                     |
+-+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     .                                     |
|                                     .                                     |
|                                     .                                     |
+-+-----+-----+-----+-----+-----+-----+-----+-----+
|                                     Source Address [N]                                     |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

Please see [RFC3376] for the definitions of the existing fields.

The new flag defined in the IGMPv3 Query is:

A : 'Acknowledgement Capable Querier' flag. Setting this flag indicates that the querier is capable of sending acknowledgements to membership reports to multicast listeners.

3.1.2. Host Behaviour

This section describes the new behaviour of IGMP hosts implementing the mechanism as specified in this draft.

1. By default, an implementation of IGMP adhering to this draft MUST enable the new listener behaviour as specified below. And if configured to do so, it MAY also fall back to operate as per [RFC3376] specification. This provision also ensures that the an existing IGMPv3 implementation can operate with a querier implementing this draft.
2. A multicast listener adhering to this new specification MUST send its group membership reports with the R-flag set, and A-bit clear, in order to indicate its requirement that the querier acknowledges its group membership reports
3. If a multicast listener sends a group membership report message with an unspecified source IP address, it MUST not send the reports with the R-flag set.

4. As soon as a multicast listener receives a general query with the A-flag cleared, it MUST fall back to normal IGMPv3 operation as per [RFC3376] specification, i.e. all its membership reports must be sent with both R-flag and A-flag cleared, and it can only retransmit its messages (RV-1) number of times. This ensures that the new implementation of IGMP listener can inter-operate with existing IGMP querier implementations.
5. The multicast listener MUST keep retransmitting its membership report until:
 - a. It receives an acknowledgement message from the querier or
 - b. It detects that there are no querier present on the link. The listener will assume this situation if it has received no queries for a 'Querier Live Time' period since it transmitted the original membership report message. The Querier Live Time is calculated as 2.5 times the Query Interval time.
6. The retransmission of the group membership reports MUST be done at random intervals within the range (0, [Unsolicited Report Interval]).
7. If a multicast listener's interface state changes, the retransmission of the previous membership report messages MUST stop, because the state machine will send new membership report messages reflecting the state change.
8. A multicast listener MUST process only valid acknowledgement messages. Invalid messages MUST be dropped silently. A valid acknowledgement message MUST obey the following rules:
 - a. The destination IP address must be equal to the IP address of the receiving interface, and cannot be a multicast, broadcast nor unspecified IP address.
 - b. The IGMP type is set to 0x23.
 - c. The checksum must be correct.
 - d. The listener is able to match the acknowledgement message with the member report message waiting for an acknowledgement.
9. If a multicast listener send multicast membership reports consisting only of group records indicating that the host no longer wish to receive multicast data for the groups specified, such as BLOCK_OLD_SOURCES record, the report MAY be sent without the R-bit set. Note that by doing this, if the report is dropped then the DR will not send multicast leave to its upstream router and may result in unnecessary multicast data forwarding.

3.1.3. Router Behaviour

This section describes the new behaviour of IGMP routers implementing the mechanism as specified in this draft.

1. By default, a querier adhering to this draft MUST send queries with the A-flag set. The implementation SHOULD also enable operators to turn the new behaviour off administratively.
2. If a querier receives a membership report message it MUST perform the same report message validation as specified in [RFC3376], i.e. that all invalid report messages MUST be silently discarded. And in addition, it MUST also perform the following checks:
 - a. If the R-flag is set, check that the source IP address is not the unspecified IP address (0.0.0.0).
3. If a querier receives a valid membership report message with the R-flag set, it MUST send an acknowledgement by transmitting an acknowledgement packet whose content is the same as the membership report message, with the following modifications:
 - a. The source IP address MUST be set to the that of the receiving interface.
 - b. The destination IP address MUST be set to the source IP address contained in the original message.
 - c. The IP checksum is recalculated.
 - d. The IGMP packet type is set to 0x23 (to be assigned by IANA).
 - e. The IGMP checksum is recalculated.
4. If a querier receives a valid membership report message with the R-flag cleared, then it will just process the report message as per [RFC3376] specification.
5. When a querier sends a query, it MUST set the A flag, unless it is configured not to do so by the administrator.

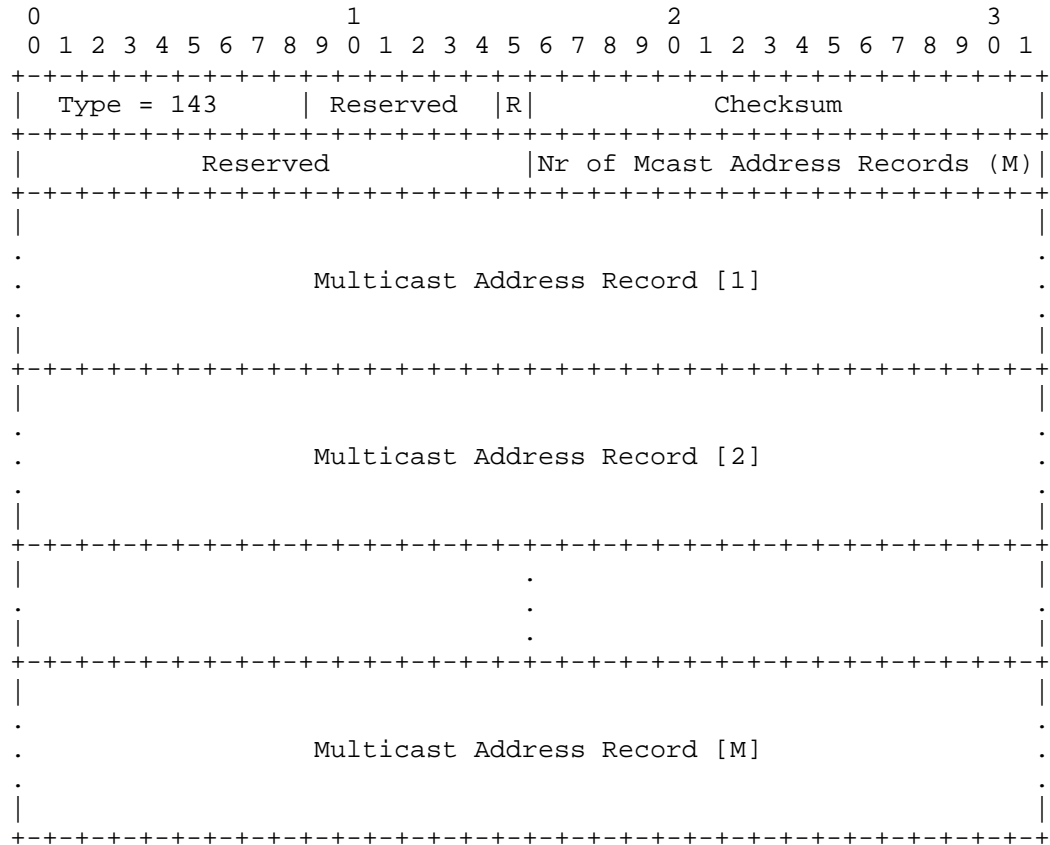
3.1.4. Backward Compatibility

To maintain compatibility with older version of IGMP implementations, both the listener and querier MUST fall back to the operation as per [RFC3376] specification, when the presence of older IGMP implementation is detected on the same network.

3.2. Resilient MLD

3.2.1. Message Formats

MLDv2 Membership Report



Please see [RFC3810] for the definitions of the existing fields.

The new flags defined in the membership report message are:

R: 'Request for Acknowledgement' flag. Setting this flag indicates that the multicast listener sending the membership report requires an acknowledgement message in reply to the current membership report message.

MLDv2 Membership Report Acknowledgement

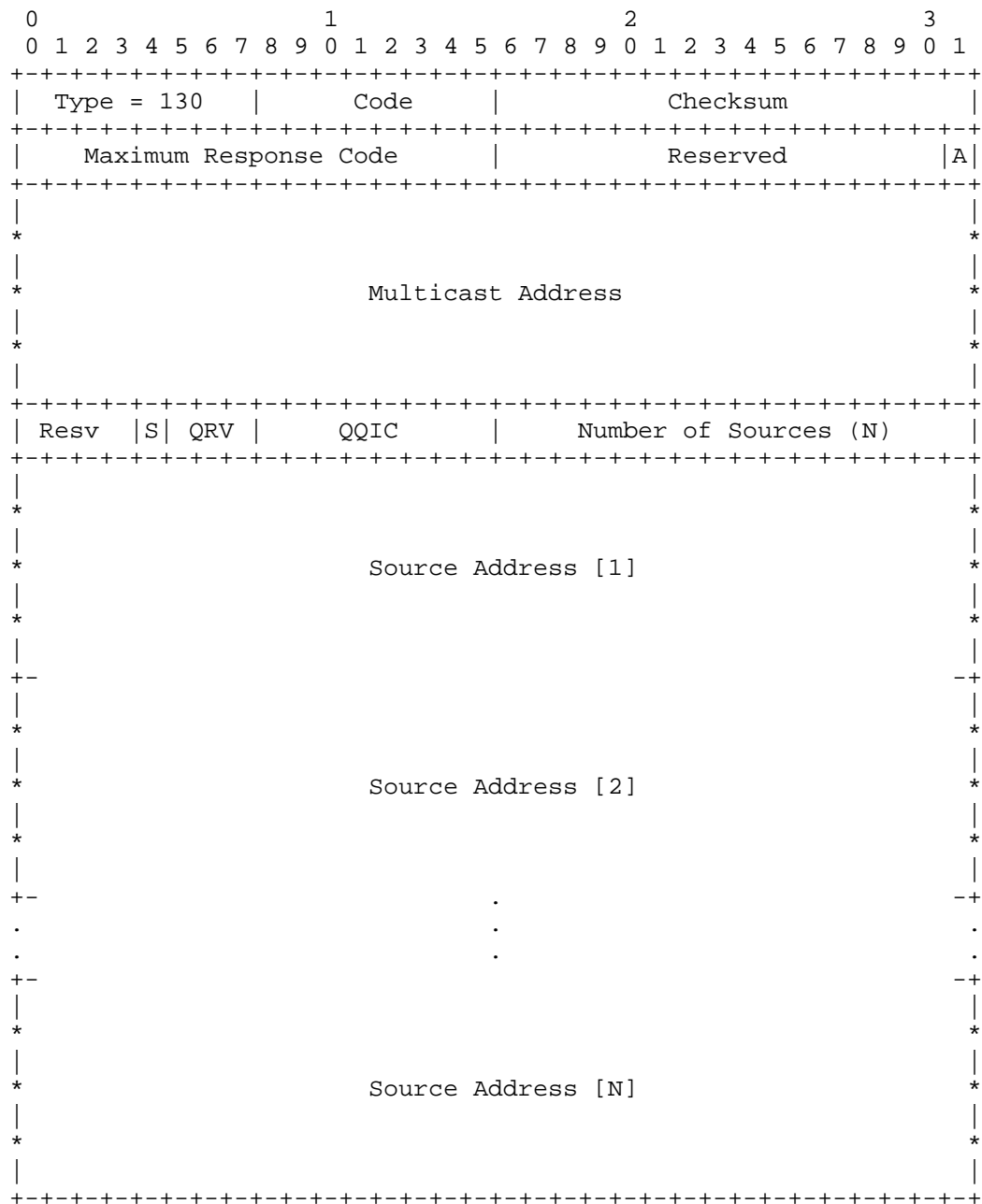
```

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-----+-----+-----+-----+-----+-----+-----+-----+
|  Type = 160      | Reserved |R|          Checksum              |
+-----+-----+-----+-----+-----+-----+-----+-----+
|          Reserved          |Nr of Mcast Address Records (M)|
+-----+-----+-----+-----+-----+-----+-----+-----+
|
|
|          Multicast Address Record [1]
|
+-----+-----+-----+-----+-----+-----+-----+-----+
|
|
|          Multicast Address Record [2]
|
+-----+-----+-----+-----+-----+-----+-----+-----+
|
|          .
|          .
|          .
+-----+-----+-----+-----+-----+-----+-----+-----+
|
|
|          Multicast Address Record [M]
|
+-----+-----+-----+-----+-----+-----+-----+-----+

```

All the fields in the MLDv2 Membership Report Acknowledgement message have the same specifications as in MLDv2 Membership Report message, except the type code is 160 (to be requested to IANA). The format is chosen so that it is very convenient for the DR to acknowledge membership report messages by simply copying the original membership report message, and modifying the copied packet with new IPv6 source and destination addresses, new ICMPv6 packet type, and recalculating the MLDv2 checksum.

The membership report acknowledgement is sent unicast to the source of the original membership report message being acknowledged.



Please see [RFC3810] for the definitions of the existing fields.

The new flag defined in MLDv2 Query is:

A : 'Acknowledgement Capable Querier' flag. Setting this flag indicates that this querier is capable of sending an acknowledgement to membership reports toward multicast listeners.

3.2.2. Host Behaviour

This section describes the new behaviour of MLD hosts implementing the mechanism as specified in this draft.

1. By default, an implementation of MLD adhering to this draft MUST enable the new listener behaviour as specified below. And if configured to do so, it MAY also fall back to operate as per [RFC3810] specification. This also ensures that existing MLDv2 implementations can inter-operate with queriers implementing this draft.
2. A multicast listener adhering to this new specification MUST send a group membership reports with the R-flag set, and the A-bit clear, indicating its wish that the querier acknowledges the listener's group membership reports.
3. If a multicast listener sends a membership report message with an unspecified source IPv6 address (::), it MUST not send the reports with the R-flag set.
4. As soon as a multicast listener receives a general query with the A-flag cleared, it MUST fall back to normal MLDv2 operation as per [RFC3810] specification, i.e. all its membership reports must be sent with both R-flag and A-flag cleared, and it can only retransmit its messages (RV-1) number of times. This ensures that the new implementation of MLD listener can inter-operate with existing MLD querier implementations.
5. The multicast listener MUST keep retransmitting the membership report until:
 - a. It receives an acknowledgement message from the querier or
 - b. It detects that there are no querier present on the link. The listener will assume this situation if it has receives no queries for 'Querier Live Time' period since it transmitted the original membership report message. The Querier Live Time is calculated as 2.5 times the Query Interval time.
6. The retransmission of the group membership reports MUST be done at random intervals within the range (0, [Unsolicited Report Interval]).
7. If a multicast listener's interface state changes, retransmission of the previous membership report message MUST stop, because the

state machine will send a new membership report message reflecting the state change.

8. A multicast listener MUST process only valid acknowledgement messages. Invalid messages MUST be dropped silently. A valid acknowledgement message MUST obey the following rules:
 - a. The destination IPv6 address MUST be equal to the IPv6 address of the receiving interface, and MUST NOT be a multicast, broadcast nor the unspecified IPv6 address (::).
 - b. The ICMPv6 packet type is set to 160.
 - c. The checksum must be correct.
 - d. The listener is able to match the acknowledgement message with the member report message waiting for an acknowledgement.
9. If a multicast listener send multicast membership reports consisting only of group records indicating that the host no longer wish to receive multicast data for the groups specified, such as BLOCK_OLD_SOURCES record, the report MAY be sent without the R-bit set. Note that by doing this, if the report is dropped then the DR will not send multicast leave to its upstream router and may result in unnecessary multicast data forwarding.

3.2.3. Router Behaviour

This section describes the new behaviour of MLD routers implementing the mechanism as specified in this draft.

1. By default, querier implementation adhering to this draft MUST send queries with the A-flag set. The implementation SHOULD also enable operators to turn the new behaviour off administratively.
2. If a querier receives a membership report message it MUST perform the same report message validation as specified in [RFC3810]. All invalid report messages MUST be silently discarded. It MUST also do the following checks:
 - a. Ensure that the A-flag is clear.
 - b. Check that if the R-flag is set, the source IPv6 address is not the unspecified IPv6 address (::).
3. If a querier receives a valid membership report message with the R-flag set, it MUST send an acknowledgement by transmitting an acknowledgement packet whose content is the same as the membership report message, with the following modifications:
 - a. The source IPv6 address MUST be set to the that of the receiving interface.
 - b. The destination IPv6 address MUST be set to the source IPv6 address contained in the original message.
 - d. The ICMPv6 packet type is set to 160 (to be requested to IANA).
 - e. The MLDv2 checksum is recalculated.

4. If a querier receives a valid membership report message with the R-flag unset, then it will process the report message as per [RFC3810] specification.
5. When a querier sends a query, it MUST set the A-flag, unless it is configured not to do so by the administrator.

3.2.4. Backward Compatibility

To maintain compatibility with older version of MLD implementations, both the listener and querier MUST fall back to the operation as per [RFC3810] specification, when the presence of older MLD implementation is detected on the same network.

3.3 Operational Consideration

If there are more than one multicast routers running IGMP/MLD in a LAN, it is advisable to have all the routers configured with the same setting for the 'Acknowledgement Capable Querier' flag to avoid temporary confusion on hosts as to whether they are allowed to send report messages with the R-flag set or not, during querier election time.

4. IANA Considerations

4.1. IGMP Type Numbers

Author of this document requests the following IGMP Type Number:

IGMPv3 Packet Type	Value
-----	-----
IGMPv3 Membership Report Acknowledgement	0x23

4.2. ICMPv6 Type Numbers

Author of this document requests the following ICMPv6 Type Number:

ICMPv6 Packet Type	Value
-----	-----
MLDv2 Membership Report Acknowledgement	160

5. Security Considerations

The original specification of both IGMPv3 and MLDv2 have some defense capability against forged messages originated off-link.

5.1. On-link Query Message Forgery

Apart from the threats described in both [RFC3376] and [RFC3810], extra threat exists in the form of forged query messages with the A-flag set, while the actual querier does not send queries with the A-flag set. This would make all listeners adhering to the mechanism proposed in this draft, keep sending extra retransmissions of the report messages without receiving acknowledgement from the querier) until they see the query with the A-flag cleared from the forged querier. To mitigate this attack, listener implementations might generate a log message if queries it receives from the same querier seem to have variable setting for the A-flag.

5.2. On-link Membership Report Message Forgery

When a querier is configured to run the new implementation as specified in this draft, additional threats exist to those described in [RFC3376] and [RFC3810]. These extra threats exist in the form of forged membership report messages with the R-flag being set. This will make the querier unnecessarily send acknowledgements to the forged listener. However this attack is mitigated by the fact that the original listener will drop these messages since it does not expect to receive acknowledgement messages.

6. Acknowledgements

A special thank you goes to Stig Venaas for providing very valuable feedback and review on this document.

7. References

7.1. Normative References

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

7.2. Informative References

- [VYNCKE] E. Vyncke , P. Thubert , E. Levy-Abegnoli , A. Yourtchenko, "Why Network-Layer Multicast is Not Always Efficient At Datalink Layer", draft-vyncke-6man-mcast-not-efficient, February 2014.
- [MCBRIDE] M. McBride, C. Perkins, "Multicast Wifi Problem Statement", draft-mcbride-mboned-wifi-mcast-problem-statement, July, 2015.

Authors' Address

Hermin Anggawijaya
Allied Telesis Labs NZ, Ltd
27 Nazareth Ave
Christchurch 8024
New Zealand

Email: hermin.anggawijaya@alliedtelesis.co.nz

PIM Working Group
INTERNET-DRAFT

Intended Status: Standard Track

X. Liu
Ericsson
F. Guo
Huawei
M. Sivakumar
Cisco
P. McAllister
Metaswitch Networks
A. Peter
Juniper Networks

Expires: September 20, 2016

March 20, 2016

A YANG data model for Internet Group Management Protocol (IGMP) and
Multicast Listener Discovery (MLD)
draft-guo-pim-igmp-mld-yang-01.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) 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/lid-abstracts.html>

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

Copyright Notice

Copyright (c) 2016 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. 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 and MLD Configuration.....	4
3.2. IGMP and MLD Operational State.....	6
4. Security Considerations.....	28
5. IANA Considerations	28
6. Acknowledgements	28
7. References	28
7.1. Normative References.....	28
7.2. Informative References.....	29
Authors' Addresses	29

1. Introduction

YANG [RFC6020] [RFC6087] 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 draft YANG data model that can be used to configure and manage Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) devices. Currently this model is incomplete, but it 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].

This draft 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 representation of some of extension features is not completely specified in this draft of the data model. This model is being circulated in its current form for early oversight and review of the basic hierarchy.

The operational state fields and rpcs of this model are also incomplete, though the structure of what has been written may be taken as representative of the structure of the model when complete.

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 covered by future Internet Drafts.

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 draft 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 draft 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 and MLD Configuration

The IGMP and MLD modules define the routing-instance-wide configuration options in a three-level hierarchy as listed below:

Global level: IGMP MLD configuration attributes for the entire routing instance

Interface-global: IGMP MLD configuration attributes applicable to all interfaces
IGMP MLD configuration attributes applied to interfaces whose interface level attributes are not existing, with same attributes' value for those

Interface-level: IGMP MLD configuration 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. The module structure also applies, where applicable, to the operational state and notifications as well.

Our current direction is to agree to a routing-instance-centric (VRF) model as opposed to protocol-centric mainly because it fits well into the routing-instance model, and it is easier to map from the VRF-centric to the protocol-centric than the other way around due to forward references.

The IGMP and MLD model will augment "/rt:routing/rt:routing-instance/rt:routing-protocols:" as opposed to augmenting "' rt:routing-instance/rt:routing-protocols:/rt:routing-protocol" as the latter would allow multiple protocol instances per VRF, which does not make sense for IGMP and MLD.

```
augment /rt:routing/rt:routing-instance/rt:routing-protocols:
  +--rw igmp
  |   +--rw global
  |   |   +--rw enable?          boolean {global-admin-enable}?
  |   |   +--rw max-entries?     uint32 {global-max-entries}?
  |   |   +--rw max-groups?      uint32 {global-max-groups}?
  |   +--rw interfaces
  |   |   +--rw last-member-query-interval?  uint16
  |   |   +--rw max-groups-per-interface?    uint32 {intf-max-groups}?
  |   |   +--rw query-interval?              uint16
  |   |   +--rw query-max-response-time?     uint16
  |   |   +--rw require-router-alert?        boolean {intf-require-router-alert
  |   |   }?
  |   |   +--rw robustness-variable?          uint8
  |   |   +--rw version?                      uint8
  |   +--rw interface* [interface]
  |   |   +--rw interface                    if:interface-ref
  |   |   +--rw enable?                      boolean {intf-admin-enable}?
  |   |   +--rw group-policy?                string
  |   |   +--rw immediate-leave?             empty {intf-immediate-leave}?
  |   |   +--rw last-member-query-interval?  uint16
  |   |   +--rw max-groups?                  uint32 {intf-max-groups}?
  |   |   +--rw max-group-sources?           uint32 {intf-max-group-sources}
  |   |   ?
  |   |   +--rw query-interval?              uint16
  |   |   +--rw query-max-response-time?     uint16
  |   |   +--rw require-router-alert?        s boolean {intf-require-router-a
  |   |   lert}?
  |   |   +--rw robustness-variable?          uint8
  |   |   +--rw source-policy?                string {intf-source-policy}?
  |   |   +--rw verify-source-subnet?         empty {intf-verify-source-subne
  |   |   t}?
  |   +--rw version?                        uint8
  |   +--rw join-group*                     inet:ipv4-address {intf-join-gr
  |   oup}?
  |   +--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 source-addr] {intf-static-group}?
|         |       +--rw group        inet:ipv4-address
|         |       +--rw source-addr   source-ipv4-addr-type
+--rw mld
+--rw global
|   +--rw enable?          boolean {global-admin-enable}?
|   +--rw max-entries?     uint32 {global-max-entries}?
|   +--rw max-groups?     uint32 {global-max-groups}?
+--rw interfaces
+--rw last-member-query-interval?   uint16
+--rw max-groups-per-interface?     uint32 {intf-max-groups}?
+--rw query-interval?               uint16
+--rw query-max-response-time?      uint16
+--rw require-router-alert?         boolean {intf-require-router-alert
}?
+--rw robustness-variable?          uint8
+--rw version?                      uint8
+--rw interface* [interface]
+--rw interface                     if:interface-ref
+--rw enable?                       boolean {intf-admin-enable}?
+--rw group-policy?                 string
+--rw immediate-leave?              empty {intf-immediate-leave}?
+--rw last-member-query-interval?   uint16
+--rw max-groups?                   uint32 {intf-max-groups}?
+--rw max-group-sources?            uint32 {intf-max-group-sources}
?
+--rw query-interval?               uint16
+--rw query-max-response-time?      uint16
+--rw require-router-alert?         boolean {intf-require-router-al
ert}?
+--rw robustness-variable?          uint8
+--rw source-policy?                string {intf-source-policy}?
+--rw verify-source-subnet?         empty {intf-verify-source-subne
t}?
+--rw version?                      uint8
+--rw join-group*                    inet:ipv6-address {intf-join-gr
oup}?
+--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

```

3.2. IGMP and MLD Operational State

The IGMP and MLD module contains operational state information also in a three-level hierarchy as mentioned earlier.

Global level: IGMP MLD operational state attributes for the entire routing instance

Interface-global: IGMP MLD interface level operational state attributes applied to interfaces whose interface level attributes do not exist, with same attributes' value for those interfaces

Interface-specific: IGMP MLD operational state attributes specific to the given interface.

augment /rt:routing-state/rt:routing-instance/rt:routing-protocols:

```

+--ro igmp
|   +--ro global
|   |   +--ro enable?          boolean {global-admin-enable}?
|   |   +--ro max-entries?      uint32 {global-max-entries}?
|   |   +--ro 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
|   |   +--ro interfaces
|   |   |   +--ro last-member-query-interval? uint16
|   |   |   +--ro max-groups-per-interface?   uint32 {intf-max-groups}?
|   |   |   +--ro query-interval?             uint16
|   |   |   +--ro query-max-response-time?    uint16
|   |   |   +--ro require-router-alert?       boolean {intf-require-router-alert
|   |   }?
|   |   |   +--ro robustness-variable?        uint8
|   |   |   +--ro version?                     uint8
|   |   |   +--ro interface* [interface]
|   |   |   |   +--ro interface                if:interface-ref
|   |   |   |   +--ro enable?                  boolean {intf-admin-enable}?
|   |   |   |   +--ro group-policy?            string
|   |   |   |   +--ro immediate-leave?        empty {intf-immediate-leave}?
|   |   |   |   +--ro last-member-query-interval? uint16
|   |   |   |   +--ro max-groups?              uint32 {intf-max-groups}?
|   |   |   |   +--ro max-group-sources?       uint32 {intf-max-group-sources}
|   |   |   }?
|   |   |   +--ro query-interval?              uint16
|   |   |   +--ro query-max-response-time?      uint16
|   |   |   +--ro require-router-alert?         boolean {intf-require-router-al
|   |   ert}?
|   |   |   +--ro robustness-variable?          uint8
|   |   |   +--ro source-policy?                string {intf-source-policy}?
|   |   |   +--ro verify-source-subnet?         empty {intf-verify-source-subne
|   |   t}?
|   |   |   +--ro version?                      uint8
|   |   |   +--ro join-group*                   inet:ipv4-address {intf-join-gr
|   |   oup}?
|   |   |   +--ro ssm-map* [source-addr group-policy] {intf-ssm-map}?

```



```

|   +--ro source-addr          ssm-map-ipv4-addr-type
|   +--ro group-policy         string
+--ro static-group* [group source-addr] {intf-static-group}?
|   +--ro group                inet:ipv4-address
|   +--ro source-addr          source-ipv4-addr-type
+--ro oper-status?             enumeration
+--ro dr?                      inet:ipv4-address
+--ro querier?                 inet:ipv4-address
+--ro joined-group*            inet:ipv4-address {intf-join-gr
oup}?

+--ro group* [address]
|   +--ro address              inet:ipv4-address
|   +--ro expire?              uint32
|   +--ro filter-mode?         enumeration
|   +--ro host-count?          uint32
|   +--ro up-time?             uint32
|   +--ro host*                inet:ipv4-address
|   +--ro last-reporter?       inet:ipv4-address
|   +--ro source* [address]
|       +--ro address          inet:ipv4-address
|       +--ro expire?          uint32
|       +--ro up-time?         uint32
|       +--ro last-reporter?   inet:ipv4-address
+--ro mld

+--ro global
|   +--ro enable?              boolean {global-admin-enable}?
|   +--ro max-entries?         uint32 {global-max-entries}?
|   +--ro 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
+--ro interfaces
|   +--ro last-member-query-interval? uint16
|   +--ro max-groups-per-interface?   uint32 {intf-max-groups}?
|   +--ro query-interval?             uint16
|   +--ro query-max-response-time?    uint16
|   +--ro require-router-alert?       boolean {intf-require-router-alert
}?

+--ro robustness-variable?          uint8
+--ro version?                      uint8

```

```

    +--ro interface* [interface]
      +--ro interface          if:interface-ref
      +--ro enable?            boolean {intf-admin-enable}?
      +--ro group-policy?      string
      +--ro immediate-leave?   empty {intf-immediate-leave}?
      +--ro last-member-query-interval? uint16
      +--ro max-groups?        uint32 {intf-max-groups}?
      +--ro max-group-sources? uint32 {intf-max-group-sources}
?
      +--ro query-interval?    uint16
      +--ro query-max-response-time? uint16
      +--ro require-router-alert? boolean {intf-require-router-al
ert}?
      +--ro robustness-variable? uint8
      +--ro source-policy?     string {intf-source-policy}?
      +--ro verify-source-subnet? empty {intf-verify-source-subne
t}?
      +--ro version?           uint8
      +--ro join-group*        inet:ipv6-address {intf-join-gr
oup}?
      +--ro ssm-map* [source-addr group-policy] {intf-ssm-map}?
      | +--ro source-addr      ssm-map-ipv6-addr-type
      | +--ro group-policy     string
      +--ro static-group* [group source-addr] {intf-static-group}?
      | +--ro group            inet:ipv6-address
      | +--ro source-addr      source-ipv6-addr-type
      +--ro oper-status?       enumeration
      +--ro querier?           inet:ipv6-address
      +--ro joined-group*      inet:ipv6-address {intf-join-gr
oup}?
      +--ro group* [address]
        +--ro address          inet:ipv6-address
        +--ro expire?          uint32
        +--ro filter-mode?     enumeration
        +--ro host-count?     uint32
        +--ro up-time?         uint32
        +--ro host*            inet:ipv6-address
        +--ro last-reporter?   inet:ipv6-address
        +--ro source* [address]
          +--ro address        inet:ipv6-address
          +--ro expire?        uint32
          +--ro up-time?       uint32
          +--ro last-reporter? inet:ipv6-address

```

3.3. IGMP and MLD RPC

```

rpcs:
+---x clear-igmp-groups {rpc-clear-groups}?
|   +---w input
|   |   +---w routing-instance? rt:routing-instance-ref
|   |   +---w interface?       leafref
|   |   +---w group?           inet:ipv4-address
+---x clear-mlld-groups {rpc-clear-groups}?
    +---w input
      +---w routing-instance? rt:routing-instance-ref
      +---w interface?       leafref
      +---w group?           inet:ipv4-address

```

4. IGMP and MLD YANG Modules

```
module ietf-igmp-mlld {
  namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mlld";
  // replace with IANA namespace when assigned
  prefix igmp-mlld;

  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>

    Editors:    ";

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

  revision 2016-03-01 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for IGMP";
  }

  /*
  * 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 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 in the routing instance;
             false to disable IGMP in the routing instance.";
    }

    leaf max-entries {
        if-feature global-max-entries;
        type uint32;
        description
            "The maximum number of entries in IGMP.";
    }

    leaf max-groups {
        if-feature global-max-groups;
        type uint32;
        description
            "The maximum number of groups that IGMP can join.";
    }
} // global-config-attributes
```

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

  leaf entries-count {
    type uint32;
    description
      "The number of entries in IGMP.";
  }
  leaf groups-count {
    type uint32;
    description
      "The number of groups that IGMP can join.";
  }
}

container statistics {
  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 interfaces whose
    per interface attributes are not existing.";

    leaf last-member-query-interval {
        type uint16 {
            range "1..65535";
        }
        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 can join.";
    }
    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;
        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
            "";
    }
    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 version.";
        reference "RFC1112, RFC2236, RFC3376.";
    }
} // interfaces-config-attributes

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

    uses interface-config-attributes-igmp-mlld;

    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 "";
        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.";
    }
}

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: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;
        description
            "true to enable IGMP on the interface;
            false to disable IGMP on the interface.";
    }
    leaf group-policy {
        type string;
        description
            "Name of the access policy used to filter IGMP membership.";
    }
    leaf immediate-leave {
        if-feature intf-immediate-leave;
        type empty;
        description
            "If present, IGMP perform an immediate leave upon receiving an
            IGMP Version 2 (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
            off the group.";
    }
    leaf last-member-query-interval {
```

```
    type uint16 {
      range "1..65535";
    }
    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 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;
    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;
    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
      "";
  }
  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.";
  }
  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 version {
    type uint8 {
      range "1..3";
    }
    description "IGMP version.";
    reference "RFC1112, RFC2236, RFC3376.";
  }
} // interface-config-attributes-igmp-mld

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

  uses interface-config-attributes-igmp-mld;

  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 "";
    leaf source-addr {
      type ssm-map-ipv6-addr-type;
      description
        "Multicast source IP address.";
    }
    leaf group-policy {
      type string;
      description
        "Name of the access policy used to filter IGMP
         membership.";
    }
  }
}

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 IP address.";
      }
      leaf source-addr {
        type source-ipv6-addr-type;
        description
          "Multicast source IP address.";
      }
    }
  } // interface-config-attributes-mlt

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

  uses interface-state-attributes-igmp-mlt;

  leaf dr {
    type inet:ipv4-address;
    description "";
  }
  leaf querier {
    type inet:ipv4-address;
    description "";
  }
  leaf-list joined-group {
    if-feature intf-join-group;
    type inet:ipv4-address;
    description
      "The routers that joined this multicast group.";
  }

  list group {
    key "address";
    description "";

    leaf address {
      type inet:ipv4-address;
      description
        "";
    }
    uses interface-state-group-attributes-igmp-mlt;
    leaf-list host {
      type inet:ipv4-address;
      description
        "";
    }
    leaf last-reporter {
      type inet:ipv4-address;
      description "";
    }
  }
  list source {
    key "address";
    description "";

    leaf address {
```

```
        type inet:ipv4-address;
        description "";
    }
    uses interface-state-source-attributes-igmp-mld;
    leaf last-reporter {
        type inet:ipv4-address;
        description "";
    }
    } // 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.";
            }
        }
        description "";
    }
} // 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;
        description "";
    }
    leaf-list joined-group {
        if-feature intf-join-group;
        type inet:ipv6-address;
        description
            "The routers that joined this multicast group.";
    }

    list group {
        key "address";
        description "";

        leaf address {
            type inet:ipv6-address;
            description
                "";
        }
    }
}
```

```

    }
    uses interface-state-group-attributes-igmp-mld;
    leaf-list host {
        type inet:ipv6-address;
        description
            "";
    }
    leaf last-reporter {
        type inet:ipv6-address;
        description "";
    }
    list source {
        key "address";
        description "";

        leaf address {
            type inet:ipv6-address;
            description "";
        }
        uses interface-state-source-attributes-igmp-mld;
        leaf last-reporter {
            type inet:ipv6-address;
            description "";
        }
    } // list source
} // list group
} // interface-state-attributes-mld

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

    leaf expire {
        type uint32;
        units seconds;
        description "";
    }
    leaf filter-mode {
        type enumeration {
            enum "include" {
                description
                    "";
            }
            enum "exclude" {
                description
                    "";
            }
        }
        description "";
    }
    leaf host-count {
        type uint32;
        description "";
    }
    leaf up-time {

```

```
        type uint32;
        units seconds;
        description "";
    }
} // interface-state-group-attributes-igmp-mld

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

    leaf expire {
        type uint32;
        units seconds;
        description "";
    }
    leaf up-time {
        type uint32;
        units seconds;
        description "";
    }
} // interface-state-source-attributes-igmp-mld

/*
 * Configuration data nodes
 */
augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols" {
    description
        "IGMP and MLD augmentation to routing instance configuration.";

    container igmp {
        description
            "IGMP configuration data.";

        container global {
            description
                "Global attributes.";
            uses global-config-attributes;
        }

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

            uses interfaces-config-attributes {
                if-feature global-interface-config;
            }

            list interface {
                key "interface";
                description
                    "List of IGMP interfaces.";
                leaf interface {
                    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;
      }
    } // interface
  } // interfaces
} // igmp

container mld {
  description
    "MLD configuration data.";

  container global {
    description
      "Global attributes.";
    uses global-config-attributes;
  }

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

    uses interfaces-config-attributes {
      if-feature global-interface-config;
    }
  }

  list interface {
    key "interface";
    description
      "List of MLD interfaces.";
    leaf interface {
      type if:interface-ref;
      must "/if:interfaces/if:interface[if:name = current()]/"
        + "ip:ipv6" {
        description
          "The interface must have IPv4 enabled.";
      }
    }
    description
      "Reference to an entry in the global interface
       list.";
  }
  uses interface-config-attributes-mld {
    if-feature per-interface-config;
  }
} // interface
} // interfaces
} // mld
} // augment
```

```
/*
 * Operational state data nodes
 */
augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols" {
  description
    "IGMP and MLD augmentation to routing instance state.";

  container igmp {
    description
      "IGMP configuration 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 {
        if-feature global-interface-config;
      }

      list interface {
        key "interface";
        description
          "List of IGMP interfaces.";
        leaf interface {
          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

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

    list interface {
        key "interface";
        description
            "List of MLD interfaces.";
        leaf interface {
            type if:interface-ref;
            must "/if:interfaces/if:interface[if:name = current()]/"
                + "ip:ipv6" {
                description
                    "The interface must have IPv4 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 tables.";

    input {
        leaf routing-instance {
            type rt:routing-instance-ref;
            description
                "Routing instance name identifying a specific routing
                instance.
                This leaf is optional for the rpc.
                If it is specified, the rpc will clear groups in the
                specified routing instance;
                if it is not specified, the rpc will clear all groups in
                all routing instances.";
        }
    }
}

```

```
    }
    leaf interface {
      type leafref {
        path "/rt:routing/rt:routing-instance"
          + "[rt:name=current()/../routing-instance]/"
          + "rt:routing-protocols/igmp/interfaces/interface/"
          + "interface";
      }
      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 IP address.
        If it is not specified, all IGMP group tables are
        cleared.";
    }
  }
} // rpc clear-igmp-groups

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

  input {
    leaf routing-instance {
      type rt:routing-instance-ref;
      description
        "Routing instance name identifying a specific routing
        instance.
        This leaf is optional for the rpc.
        If it is specified, the rpc will clear groups in the
        specified routing instance;
        if it is not specified, the rpc will clear all groups in
        all routing instances.";
    }
  }
  leaf interface {
    type leafref {
      path "/rt:routing/rt:routing-instance"
        + "[rt:name=current()/../routing-instance]/"
        + "rt:routing-protocols/mld/interfaces/interface/"
        + "interface";
    }
    description
      "Name of the MLD interface.
      If it is not specified, groups from all interfaces are
      cleared.";
  }
  leaf group {
    type inet:ipv4-address;
    description
      "Multicast group IP address.
```

```
        If it is not specified, all MLD group tables are
        cleared.";
    }
}
} // rpc clear-mld-groups

/*
 * Notifications
 */
}
```

4. Security Considerations

The data model defined does not introduce any security implications. This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

5. IANA Considerations

TBD

6. Acknowledgements

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

7. References

7.1. 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>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.
- [I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-05 (work in progress), October 2015.

7.2. 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.
- [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, August 2006.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, August 2006.

Authors' Addresses

Xufeng Liu
Ericsson
1595 Spring Hill Road, Suite 500
Vienna VA 22182
USA

Email: xufeng.liu@ericsson.com

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

Email: guofeng@huawei.com

Mahesh Sivakumar
Cisco Systems, Inc.
510 McCarthy Blvd
Milpitas, California 95035
United States

Email: masivaku@cisco.com

Pete McAllister

Metaswitch Networks
100 Church Street
Enfield EN2 6BQ
UK

EMail: pete.mcallister@metaswitch.com

Anish Peter
Juniper Networks
Electra, Exora Business Park
Bangalore, KA 560103
India

EMail: anishp@juniper.net

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: June 8, 2017

J. Arango
S. Venaas
Cisco Systems
I. Kouvelas
Arista Networks Inc.
D. Farinacci
lispers.net
December 5, 2016

PIM Join Attributes for LISP Environments
draft-ietf-pim-join-attributes-for-lisp-06.txt

Abstract

This document defines two PIM Join/Prune attributes that support the construction of multicast distribution trees where the root and receivers are located in different LISP sites. These attributes allow the receiver site to select between unicast and multicast underlay transport and to convey the RLOC (Routing Locator) address of the receiver ETR (Egress Tunnel Router) to the control plane of the root ITR (Ingress Tunnel Router).

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 June 8, 2017.

Copyright Notice

Copyright (c) 2016 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. Requirements Notation	3
3. PIM Join/Prune Attributes	3
4. The Transport Attribute	4
4.1. Transport Attribute Format	4
4.2. Using the Transport Attribute	5
5. Receiver ETR RLOC Attribute	5
5.1. Receiver RLOC Attribute Format	6
5.2. Using the Receiver RLOC Attribute	6
6. Security Considerations	7
7. IANA Considerations	7
8. References	7
8.1. Normative References	7
8.2. Informative References	8
Authors' Addresses	8

1. Introduction

The construction of multicast distribution trees where the root and receivers are located in different LISP sites [RFC6830] is defined in [RFC6831]. Creation of (root-EID,G) state in the root site requires that unicast LISP-encapsulated Join/Prune messages be sent from an ETR on the receiver site to an ITR on the root site. The term EID is short for Endpoint ID.

[RFC6831] specifies that (root-EID,G) data packets are to be LISP-encapsulated into (root-RLOC,G) multicast packets. However, a wide deployment of multicast connectivity between LISP sites is unlikely to happen any time soon. In fact, some implementations are initially focusing on unicast transport with head-end replication between root and receiver sites.

The unicast LISP-encapsulated Join/Prune message specifies the (root-EID,G) state that needs to be established in the root site, but conveys nothing about the receivers capability or desire to use multicast as the underlying transport. This document specifies a Join/Prune attribute that allows the receiver ETR to select the desired transport.

The term transport in this document is intentionally somewhat vague. Currently it is used just to indicate whether multicast or head-end replication is used. Which means that the outer destination address is either a unicast or multicast address. Future documents may specify how other types of delivery, encapsulation or underlay are used.

Knowledge of the receiver ETR's RLOC address is also essential to the control plane of the root ITR. The RLOC address determines the downstream destination for unicast head-end replication and identifies the receiver ETR that needs to be notified should the root ITR of the distribution tree move to another site. The root ITR can change when the source EID is roaming to another LISP site.

Service providers may implement uRPF policies requiring that the outer source address of the LISP-encapsulated Join/Prune message be the address of the receiver ETR's core-facing interface used to physically transmit the message. However, due to policy and load balancing considerations, the outer source address may not be the RLOC on which the receiver site wishes to receive a particular flow. This document specifies a Join/Prune attribute that conveys the appropriate receiver ETR's RLOC address to the control plane of the root ITR.

This document uses terminology defined in [RFC6830], such as EID, RLOC, ITR and ETR.

2. Requirements Notation

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].

3. PIM Join/Prune Attributes

PIM Join/Prune attributes are defined in [RFC5384] by introducing a new Encoded-Source type that, in addition to the Join/Prune source, can carry multiple type-length-value (TLV) attributes. These attributes apply to the individual Join/Prune sources on which they are stored.

The attributes defined in this document conform to the format of the encoding type defined in [RFC5384]. The attributes would typically be the same for all the sources in the Join/Prune message. Hence we RECOMMEND using the hierarchical Join/Prune attribute scheme defined in [RFC7887]. This hierarchical system allows attributes to be conveyed on the Upstream Neighbor Address field, thus enabling the

efficient application of a single attribute instance to all the sources in the Join/Prune message.

LISP xTRs do not exchange PIM Hello Messages and hence no Hello option is defined to negotiate support for these attributes. Systems that support unicast head-end replication are assumed to support these attributes.

4. The Transport Attribute

It is essential that a mechanism be provided by which the desired transport can be conveyed by receiver sites. Root sites with multicast connectivity will want to leverage multicast replication. However, not all receiver sites can be expected to have multicast connectivity. It is thus desirable that root sites be prepared to support (root-EID,G) state with a mixture of multicast and unicast output state. This document specifies a Join/Prune attribute that allows the receiver to select the desired underlying transport.

4.1. Transport Attribute Format

```

      0                               1                               2
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3
+-----+-----+-----+-----+-----+-----+-----+-----+
|F|E| Type = TBD| Length = 1      | Transport      |
+-----+-----+-----+-----+-----+-----+-----+-----+

```

F-bit: The Transitive bit. Specifies whether the attribute is transitive or non-transitive. MUST be set to zero. This attribute is ALWAYS non-transitive.

E-bit: End-of-Attributes bit. Specifies whether this attribute is the last. Set to zero if there are more attributes. Set to 1 if this is the last attribute.

Type: The Transport Attribute type is TBD.

Length: The length of the Transport Attribute value. MUST be set to 1.

Transport: The type of transport being requested. Set to 0 for multicast. Set to 1 for unicast. The values from 2 to 255 may be assigned in the future.

4.2. Using the Transport Attribute

Hierarchical Join/Prune attribute instances [RFC7887] SHOULD be used when the same Transport Attribute is to be applied to all the sources within the Join/Prune message or all the sources within a group set. The root ITR MUST accept Transport Attributes in the Upstream Neighbor Encoded-Unicast address, Encoded-Group addresses, and Encoded-Source addresses.

There MUST NOT be more than one Transport Attribute within the same encoded address. If an encoded address has more than one instance of the attribute, the root ITR MUST discard all affected Join/Prune sources. The root ITR MUST also discard all affected Join/Prune sources if the transport attribute value is unknown.

5. Receiver ETR RLOC Attribute

When a receiver ETR requests unicast head-end replication for a given (root-EID,G) entry, the PIM control plane of the root ITR must maintain an output interface list ("oif-list") entry for the receiver ETR and its corresponding RLOC address. This allows the root ITR to perform unicast LISP-encapsulation of multicast data packets to each and every receiver ETR that has requested unicast head-end replication.

The PIM control plane of the root ITR could potentially determine the RLOC address of the receiver ETR from the outer source address field of LISP-encapsulated Join/Prune message. However, receiver ETRs are subject to uRPF checks by the network providers on each core-facing interface. The outer source address must therefore be the RLOC of the core-facing interface used to physically transmit the LISP-encapsulated Join/Prune message. Due to policy and load balancing considerations, that may not be the RLOC on which the receiver site wishes to receive a particular flow. This document specifies a Join/Prune attribute that conveys the appropriate receiver RLOC address to the PIM control plane of the root ITR.

To support root-EID mobility, receiver ETRs must also be tracked by the LISP control plane of the root ITR, regardless of the underlying transport. When the root-EID moves to a new root ITR in a different LISP site, the receiver ETRs do not know the root-EID has moved and therefore do not know the RLOC of the new root ITR. This is true for both unicast and multicast transport modes. The new root ITR does not have any receiver ETR state. Therefore, it is the responsibility of the old root ITR to inform the receiver ETRs that the root-EID has moved. When the old root ITR detects that the root-EID has moved, it sends a LISP SMR message to each receiver ETR. The receiver ETRs do a mapping database lookup to retrieve the RLOC of the new root ITR.

The old root ITR detects that the root-EID has moved when it receives a Map-Notify from the Map-Server. The transmission of the Map-Notify is triggered when the new root ITR registers the root-EID [I-D.portoles-lisp-eid-mobility]. When a receiver ETR determines that the root ITR has changed it will send a LISP-encapsulated PIM prune message to the old root XTR and a LISP-encapsulated PIM join message to the new root XTR.

5.1. Receiver RLOC Attribute Format

```

      0               1               2               3
      0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
|F|E|Type=TBD+1 |   Length   |  Addr Family  | Receiver RLOC
+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+...
```

F-bit: The Transitive bit. Specifies whether this attribute is transitive or non-transitive. MUST be set to zero. This attribute is ALWAYS non-transitive.

E-bit: End-of-Attributes bit. Specifies whether this attribute is the last. Set to zero if there are more attributes. Set to 1 if this is the last attribute.

Type: The Receiver RLOC Attribute type is TBD+1.

Length: The length in octets of the attribute value. MUST be set to the length in octets of the receiver RLOC address plus one octet to account for the Address Family field.

Addr Family: The PIM Address Family of the receiver RLOC as defined in [RFC7761].

Receiver RLOC: The RLOC address on which the receiver ETR wishes to receive the unicast-encapsulated flow.

5.2. Using the Receiver RLOC Attribute

Hierarchical Join/Prune attribute instances [RFC7887] SHOULD be used when the same Receiver RLOC attribute is to be applied to all the sources within the message or all the sources within a group set. The root ITR MUST accept Transport Attributes in the Upstream Neighbor Encoded-Unicast address, Encoded-Group addresses, and Encoded-Source addresses.

There MUST NOT be more than one Receiver RLOC Attribute within the same encoded address. If an encoded address has more than one instance of the attribute, the root ITR MUST discard all affected Join/Prune sources. The root ITR MUST also discard all affected Join/Prune sources if the address family is unknown, or the address length is incorrect for the specified address family.

6. Security Considerations

Security of Join/Prune Attributes is only guaranteed by the security of the PIM packet. The attributes specified herein do not enhance or diminish the privacy or authenticity of a Join/Prune message. A site that legitimately or maliciously sends and delivers a Join/Prune message to another site will equally be able to append these and any other attributes it wishes. See [RFC5384] for general security considerations for Join/Prune attributes.

7. IANA Considerations

Two new PIM Join/Prune attribute types need to be assigned. Type 5 is being requested for the Transport Attribute. Type 6 is being requested for the Receiver RLOC Attribute.

A registry needs to be created for the Join/Prune Transport attribute. The name of the registry should be PIM Join/Prune Transport Types. The registration policy is IETF Review, and the values are in the range 0-255. This document assigns the value 0 for multicast and 1 for unicast.

8. References

8.1. Normative References

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC5384] Boers, A., Wijnands, I., and E. Rosen, "The Protocol Independent Multicast (PIM) Join Attribute Format", RFC 5384, DOI 10.17487/RFC5384, November 2008, <<http://www.rfc-editor.org/info/rfc5384>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<http://www.rfc-editor.org/info/rfc6830>>.

- [RFC6831] Farinacci, D., Meyer, D., Zwiebel, J., and S. Venaas, "The Locator/ID Separation Protocol (LISP) for Multicast Environments", RFC 6831, DOI 10.17487/RFC6831, January 2013, <<http://www.rfc-editor.org/info/rfc6831>>.
- [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>>.
- [RFC7887] Venaas, S., Arango, J., and I. Kouvelas, "Hierarchical Join/Prune Attributes", RFC 7887, DOI 10.17487/RFC7887, June 2016, <<http://www.rfc-editor.org/info/rfc7887>>.

8.2. Informative References

- [I-D.portoles-lisp-eid-mobility]
Portoles-Comeras, M., Ashtaputre, V., Moreno, V., Maino, F., and D. Farinacci, "LISP L2/L3EID Mobility Using a Unified Control Plane", draft-portoles-lisp-eid-mobility-01 (work in progress), October 2016.

Authors' Addresses

Jesus Arango
Cisco Systems
170 Tasman Drive
San Jose, CA 95134
USA

Email: jeearango@cisco.com

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

Email: stig@cisco.com

Isidor Kouvelas
Arista Networks Inc.
5453 Great America Parkway
Santa Clara, CA 95054
USA

Email: kouvelas@arista.com

Dino Farinacci
lispers.net
San Jose, CA
USA

Email: farinacci@gmail.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: August 4, 2018

IJ. Wijnands
S. Venaas
Cisco Systems, Inc.
M. Brig
Aegis BMD Program Office
A. Jonasson
Swedish Defence Material Administration (FMV)
January 31, 2018

PIM Flooding Mechanism and Source Discovery
draft-ietf-pim-source-discovery-bsr-12

Abstract

PIM Sparse-Mode (PIM-SM) uses a Rendezvous Point (RP) and shared trees to forward multicast packets from new sources. Once last hop routers receive packets from a new source, they may join the Shortest Path Tree for the source for optimal forwarding. This draft defines a new mechanism that provides a way to support PIM-SM without the need for PIM registers, RPs or shared trees. Multicast source information is flooded throughout the multicast domain using a new generic PIM flooding mechanism. This allows last hop routers to learn about new sources without receiving initial data packets.

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 August 4, 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.

Table of Contents

1. Introduction	2
1.1. Conventions Used in This Document	4
1.2. Terminology	4
2. Testing and Deployment Experiences	4
3. A Generic PIM Flooding Mechanism	5
3.1. PFM Message Format	6
3.2. Administrative Boundaries	7
3.3. Originating PFM Messages	7
3.4. Processing PFM Messages	9
3.4.1. Initial Checks	9
3.4.2. Processing and Forwarding of PFM Messages	10
4. Distributing Source Group Mappings	10
4.1. Group Source Holdtime TLV	10
4.2. Originating Group Source Holdtime TLVs	11
4.3. Processing GSH TLVs	13
4.4. The First Packets and Bursty Sources	13
4.5. Resiliency to Network Partitioning	14
5. Configurable Parameters	14
6. Security Considerations	15
7. IANA Considerations	16
8. Acknowledgments	16
9. References	16
9.1. Normative References	16
9.2. Informative References	17
Authors' Addresses	17

1. Introduction

PIM Sparse-Mode (PIM-SM) [RFC7761] uses a Rendezvous Point (RP) and shared trees to forward multicast packets to Last Hop Routers (LHR). After the first packet is received by a LHR, the source of the multicast stream is learned and the Shortest Path Tree (SPT) can be joined. This draft defines a new mechanism that provides a way to support PIM-SM without the need for PIM registers, RPs or shared trees. Multicast source information is flooded throughout the multicast domain using a new generic PIM flooding mechanism. By

removing the need for RPs and shared trees, the PIM-SM procedures are simplified, improving router operations, management and making the protocol more robust. Also the data packets are only sent on the SPTs, providing optimal forwarding.

This mechanism has some similarities to PIM Dense Mode (PIM-DM) with its State-Refresh signaling [RFC3973], except that there is no initial flooding of data packets for new sources. It provides the traffic efficiency of PIM-SM, while being as easy to deploy as PIM-DM. The downside is that it cannot provide forwarding of initial packets from a new source, see Section 4.4. PIM-DM is very different from PIM-SM and not as mature, Experimental vs Internet Standard, and there are only a few implementations. The solution in this document consists of a lightweight source discovery mechanism on top of the Source-Specific Multicast (SSM) [RFC4607] parts of PIM-SM. It is feasible to implement only a subset of PIM-SM to provide SSM support, and in addition implement the mechanism in this draft to offer a source discovery mechanism for applications that do not provide their own source discovery.

This document defines a generic flooding mechanism for distributing information throughout a PIM domain. While the forwarding rules are largely similar to Bootstrap Router mechanism (BSR) [RFC5059], any router can originate information, and it allows for flooding of any kind of information. Each message contains one or more pieces of information encoded as TLVs (type, length and value). This document defines one TLV used for distributing information about active multicast sources. Other documents may define additional TLVs.

Note that this document is experimental. While the flooding mechanism is largely similar to BSR, there are some concerns about scale as there can be multiple routers distributing information, and potentially larger amount of data that needs to be processed and stored. Distributing knowledge of active sources in this way is new, and there are some concerns, mainly regarding potentially large amounts of source states that need to be distributed. While there has been some testing in the field, we need to learn more about the forwarding efficiency, both the amount of processing per router, and propagation delay, and the amount of state that can be distributed. In particular, how many active sources one can support without consuming too many resources. There are also parameters, see Section 5, that can be tuned regarding how frequently information is distributed, and it is not clear what parameters are useful for different types of networks.

1.1. Conventions Used in This Document

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

1.2. Terminology

RP: Rendezvous Point

BSR: Bootstrap Router

RPF: Reverse Path Forwarding

SPT: Shortest Path Tree

FHR: First Hop Router, directly connected to the source

LHR: Last Hop Router, directly connected to the receiver

PFM: PIM Flooding Mechanism

PFM-SD: PFM Source Discovery

SG Mapping: Multicast source group mapping

2. Testing and Deployment Experiences

A prototype of this specification has been implemented and there has been some limited testing in the field. The prototype was tested in a network with low bandwidth radio links. The network has frequent topology changes, including frequent link or router failures. Previously existing mechanisms like PIM-SM and PIM-DM were tested.

With PIM-SM the existing RP election mechanisms were found to be too slow. With PIM-DM, issues were observed with new multicast sources starving low bandwidth links even when there are no receivers, in some cases such that there was no bandwidth left for prune messages.

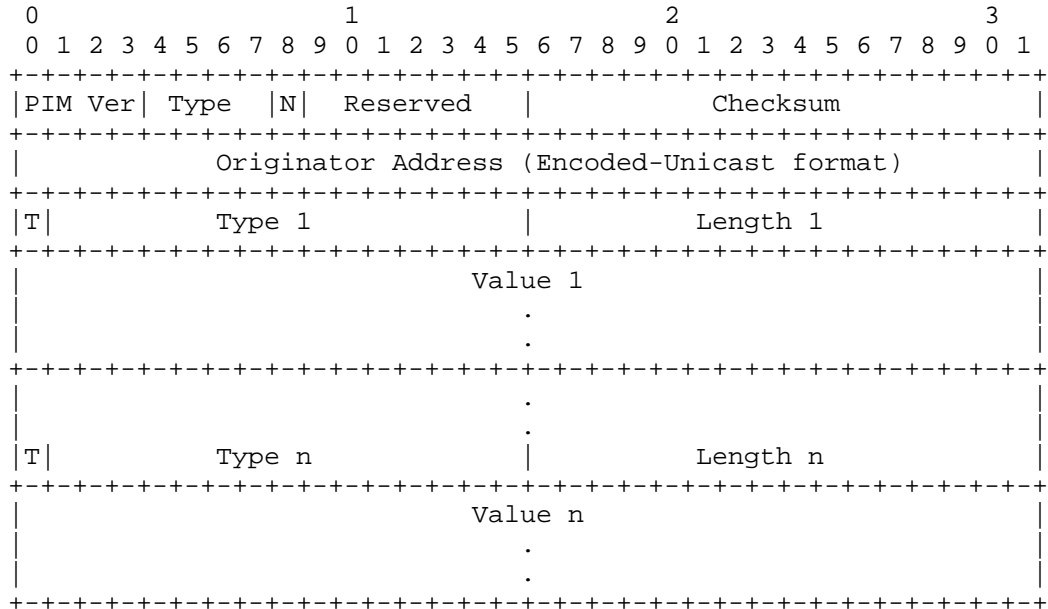
For the PFM-SD prototype tests, all routers were configured to send PFM-SD for directly connected source and to cache received announcements. Applications such as SIP with multicast subscriber discovery, multicast voice conferencing, position tracking and NTP were successfully tested. The tests went quite well. Packets were rerouted as needed and there were no unnecessary forwarding of packets. Ease of configuration was seen as a plus.

3. A Generic PIM Flooding Mechanism

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. BSR as defined, is only able to distribute Group to RP mappings. This document defines a more generic mechanism that can flood any kind of information. Administrative boundaries, see Section 3.2, may be configured to limit to which parts of a network the information is flooded.

The forwarding rules are identical to BSR, except that one can control whether routers should forward unsupported data types. For some types of information it is quite useful that it can be distributed without all routers having to support the particular type, while there may also be types where it is necessary for every single router to support it. The mechanism includes an originator address which is used for RPF checking to restrict the flooding, and prevent loops, just like BSR. Like BSR, messages are forwarded hop by hop; the messages are link-local and each router will process and resend the messages. Note that there is no equivalent to the BSR election mechanism; there can be multiple originators. This mechanism is named the PIM Flooding Mechanism (PFM).

3.1. PFM Message Format



PIM Version, Reserved and Checksum: As specified in [RFC7761].

Type: PIM Message Type. Value (pending IANA) for a PFM message.

[N]o-Forward bit: When set, this bit means that the PFM message is not to be forwarded. This bit is defined to prevent Bootstrap message forwarding in [RFC5059].

Originator Address: The address of the router that originated the message. This can be any address assigned to the originating router, but MUST be routable in the domain to allow successful forwarding. The format for this address is given in the Encoded-Unicast address in [RFC7761].

[T]ransitive bit: Each TLV in the message includes a bit called the Transitive bit that controls whether the TLV is forwarded by routers that do not support the given type. See Section 3.4.2.

Type 1..n: A message contains one or more TLVs, in this case n TLVs. The Type specifies what kind of information is in the Value. The type range is from 0 to 32767 (15 bits).

Length 1..n: The length of the the value field in octets.

Value 1..n: The value associated with the type and of the specified length.

3.2. Administrative Boundaries

PFM messages are generally forwarded hop by hop to all PIM routers. However, similar to BSR, one may configure administrative boundaries to limit the information to certain domains or parts of the network. Implementations MUST have a way of defining a set of interfaces on a router as administrative boundaries for all PFM messages, or optionally for certain TLVs, allowing for different boundaries for different TLVs. Usually one wants boundaries to be bidirectional, but an implementation MAY also provide unidirectional boundaries. When forwarding a message, a router MUST NOT send it out an interface that is an outgoing boundary, including bidirectional boundary, for all PFM messages. If an interface is an outgoing boundary for certain TLVs, the message MUST NOT be sent out the interface if it is a boundary for all the TLVs in the message. Otherwise the router MUST remove all the boundary TLVs from the message and send the message with the remaining TLVs. Also, when receiving a PFM message on an interface, the message MUST be discarded if the interface is an incoming boundary, including bidirectional boundary, for all PFM messages. If the interface is an incoming boundary for certain TLVs, the router MUST ignore all boundary TLVs. If all the TLVs in the message are boundary TLVs, then the message is effectively ignored. Note that when forwarding an incoming message, the boundary is applied before forwarding. If the message was discarded or all the TLVs were ignored, then no message is forwarded. When a message is forwarded, it MUST NOT contain any TLVs for which the incoming interface is an incoming, or bidirectional, boundary.

3.3. Originating PFM Messages

A router originates a PFM message when it needs to distribute information using a PFM message to other routers in the network. When a message is originated depends on what information is distributed. For instance this document defines a TLV to distribute information about active sources. When a router has a new active source, a PFM message should be sent as soon as possible. Hence a PFM message should be sent every time there is a new active source. However, the TLV also contains a holdtime and PFM messages need to be sent periodically. Generally speaking, a PFM message would typically be sent when there is a local state change, causing information to be distributed with PFM to change. Also, some information may need to be sent periodically. These messages are called triggered and periodic messages, respectively. Each TLV definition will need to define when a triggered PFM message needs to be originated, and also whether to send periodic messages, and how frequent.

A router MUST NOT originate more than `Max_PFM_Message_Rate` messages per minute. This document does not mandate how this should be implemented, but some possible ways could be having a minimal time between each message, counting the number of messages originated and resetting the count every minute, or using a leaky bucket algorithm. One benefit of using a leaky bucket algorithm is that it can handle bursts better. The default value of `Max_PFM_Message_Rate` is 6. The value MUST be configurable. Depending on the network, one may want to use a larger value of `Max_PFM_Message_Rate` to favor propagation of new information, but with a large number of routers and many updates, the total number of messages might become too large and require too much processing.

There MUST be a minimum of `Min_PFM_Message_Gap` milliseconds between each originated message. The default value of `Min_PFM_Message_Gap` is 1000 (1 second). The value MUST be configurable.

Unless otherwise specified by the TLV definitions, there is no relationship between different TLVs, and an implementation can choose whether to combine TLVs in one message or across separate messages. It is RECOMMENDED to combine multiple TLVs in one message, to reduce the number of messages, but it is also RECOMMENDED that the message is small enough to avoid fragmentation at the IP layer. When a triggered PFM message needs to be sent due to a state change, a router MAY send a message containing only the information that changed. If there are many changes occurring at about the same time, it might be possible to combine multiple changes in one message. In the case where periodic messages are also needed, an implementation MAY include periodic PFM information in a triggered PFM. E.g., if some information needs to be sent every 60 seconds and a triggered PFM is about to be sent 20 seconds before the next periodic PFM was scheduled, the triggered PFM might include the periodic information and the next periodic PFM can then be scheduled 60 seconds after that, rather than 20 seconds later.

When a router originates a PFM message, it puts one of its own addresses in the originator field. An implementation MUST allow an administrator to configure which address is used. For a message to be received by all routers in a domain, all the routers need to have a route for this address due to the RPF based forwarding. Hence an administrator needs to be careful which address to choose. When this is not configured, an implementation MUST NOT use a link-local address. It is RECOMMENDED to use an address of a virtual interface such that the originator can remain unchanged and routable independent of which physical interfaces or links may go down.

The No-Forward bit MUST NOT be set, except for the case when a router receives a PIM Hello from a new neighbor, or a PIM Hello with a new

Generation Identifier, defined in [RFC7761], is received from an existing neighbor. In that case an implementation MAY send PFM messages containing relevant information so that the neighbor can quickly get the correct state. The definition of the different PFM message TLVs need to specify what, if anything, needs to be sent in this case. If such a PFM message is sent, the No-Forward bit MUST be set, and the message must be sent within 60 seconds after the neighbor state change. The processing rules for PFM messages will ensure that any other neighbors on the same link ignores the message. This behavior and the choice of 60 seconds is similar to what is defined for the No-Forward bit in [RFC5059].

3.4. Processing PFM Messages

A router that receives a PFM message MUST perform the initial checks specified here. If the checks fail, the message MUST be dropped. An error MAY be logged, but otherwise the message MUST be dropped silently. If the checks pass, the contents is processed according to the processing rules of the included TLVs.

3.4.1. Initial Checks

In order to do further processing, a message MUST meet the following requirements. The message MUST be from a directly connected PIM neighbor, the destination address MUST be ALL-PIM-ROUTERS. Also, the interface MUST NOT be an incoming, nor bidirectional, administrative boundary for PFM messages, see Section 3.2. If No-Forward is not set, the message MUST be from the RPF neighbor of the originator address. If No-Forward is set, this system, the router doing these checks, MUST have enabled the PIM protocol within the last 60 seconds. See Section 3.3 for details. In pseudo-code the algorithm is as follows:

```

if ((DirectlyConnected(PFM.src_ip_address) == FALSE) OR
    (PFM.src_ip_address is not a PIM neighbor) OR
    (PFM.dst_ip_address != ALL-PIM-ROUTERS) OR
    (Incoming interface is admin boundary for PFM)) {
    drop the message silently, optionally log error.
}
if (PFM.no_forward_bit == 0) {
    if (PFM.src_ip_address !=
        RPF_neighbor(PFM.originator_ip_address)) {
        drop the message silently, optionally log error.
    }
} else if (more than 60 seconds elapsed since PIM enabled)) {
    drop the message silently, optionally log error.
}

```

Note that `src_ip_address` is the source address in the IP header of the PFM message. Originator is the originator field inside the PFM message, and is the router that originated the message. When the message is forwarded hop by hop, the originator address never changes, while the source address will be an address belonging to the router that last forwarded the message.

3.4.2. Processing and Forwarding of PFM Messages

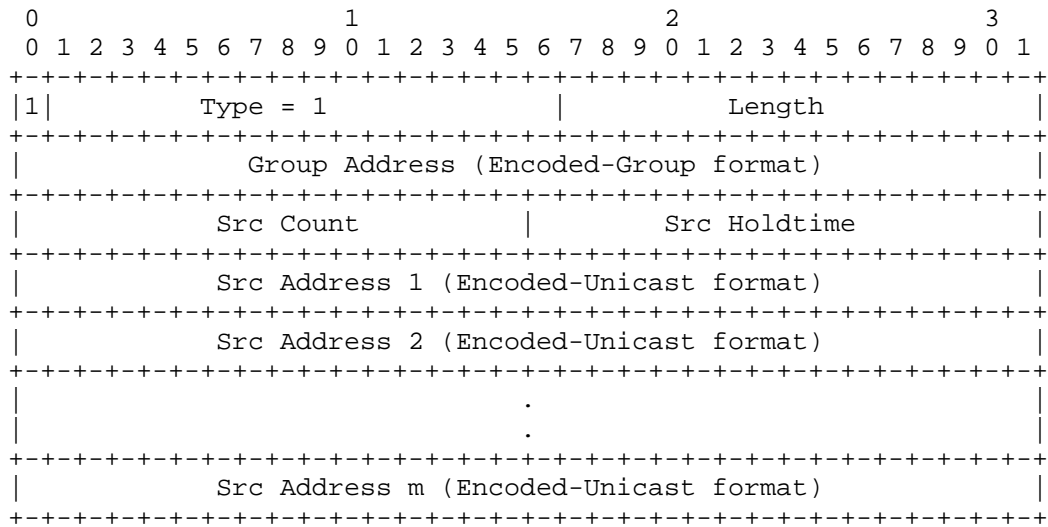
When the message is received, the initial checks above must be performed. If it passes the checks, then for each included TLV, perform processing according to the specification for that TLV.

After processing, the message is forwarded. Some TLVs may be omitted or modified in the forwarded message. This depends on administrative boundaries, see Section 3.2, the type specification and the setting of the Transitive bit for the TLV. If a router supports the type, then the TLV is forwarded with no changes unless otherwise specified by the type specification. A router not supporting the given type MUST include the TLV in the forwarded message if and only if the Transitive bit is set. Whether a router supports the type or not, the value of the Transitive bit MUST be preserved if the TLV is included in the forwarded message. The message is forwarded out of all interfaces with PIM neighbors (including the interface it was received on). As specified in Section 3.2, if an interface is an outgoing boundary for any TLVs, the message MUST NOT be sent out the interface if it is an outgoing boundary for all the TLVs in the message. Otherwise the router MUST remove any outgoing boundary TLVs of the interface from the message and send the message out that interface with the remaining TLVs.

4. Distributing Source Group Mappings

The generic flooding mechanism (PFM) defined in the previous section can be used for distributing source group mappings about active multicast sources throughout a PIM domain. A Group Source Holdtime (GSH) TLV is defined for this purpose.

4.1. Group Source Holdtime TLV



1: The Transitive bit is set to 1. This means that this type will be forwarded even if a router does not support it. See Section 3.4.2.

Type: This TLV has type 1.

Length: The length of the value in octets.

Group Address: The group that sources are to be announced for. The format for this address is given in the Encoded-Group format in [RFC7761].

Src Count: The number of source addresses that are included.

Src Holdtime: The Holdtime (in seconds) for the included source(s).

Src Address: The source address for the corresponding group. The format for these addresses is given in the Encoded-Unicast address in [RFC7761].

4.2. Originating Group Source Holdtime TLVs

A PFM message MAY contain one or more Group Source Holdtime (GSH) TLVs. This is used to flood information about active multicast sources. Each FHR that is directly connected to an active multicast source originates PFM messages containing GSH TLVs. How a multicast router discovers the source of the multicast packet and when it considers itself the FHR follows the same procedures as the registering process described in [RFC7761]. When a FHR has decided

that a register needs to be sent per [RFC7761], the SG is not registered via the PIM-SM register procedures, but the SG mapping is included in an GSH TLV in a PFM message. Note, only the SG mapping is distributed in the message, not the entire packet as would have been done with a PIM register.

The PFM messages containing the GSH TLV are sent periodically for as long as the multicast source is active, similar to how PIM registers are sent periodically. This means that as long as the source is active, it is included in a PFM message originated every Group_Source_Holdtime_Period seconds, within the general PFM timing requirements in Section 3.3. The default value of Group_Source_Holdtime_Period is 60. The value MUST be configurable. The holdtime for the source MUST be set to either zero or Group_Source_Holdtime_Holdtime. The value of the Group_Source_Holdtime_Holdtime parameter MUST be larger than Group_Source_Holdtime_Period. It is RECOMMENDED to be 3.5 times the Group_Source_Holdtime_Period. The default value is 210 (seconds). The value MUST be configurable. A source MAY be announced with a holdtime of zero to indicate that the source is no longer active.

If an implementation supports originating GSH TLVs with different holdtimes for different sources, it can if needed send multiple TLVs with the same group address. Due to the format, all the sources in the same TLV have the same holdtime.

When a new source is detected, an implementation MAY send a PFM message containing just that particular source. However, it MAY also include information about other sources that were just detected, sources that are scheduled for periodic announcement later, or other types of information. See Section 3.3 for details. Note that when a new source is detected, one should trigger sending of a PFM message as soon as possible, while if a source becomes inactive, there is no reason to trigger a message. There is no urgency in removing state for inactive sources. Note that the message timing requirements in Section 3.3 apply. This means that one cannot always send a triggered message immediately when a new source is detected. In order to meet the timing requirements, sending of the message may have to be delayed a small amount of time.

When a new PIM neighbor is detected, or an existing neighbor changes Generation Identifier, an implementation MAY send a triggered PFM message containing GSH TLVs for any Source Group mappings it has learned by receiving PFM GSH TLVs as well as any active directly connected sources. See Section 3.3 for further details.

4.3. Processing GSH TLVs

A router that receives a PFM message containing GSH TLVs MUST parse the GSH TLVs and store each of the GSH TLVs as SG mappings with a holdtimer started with the advertised holdtime, unless the implementation specifically does not support GSH TLVs, the router is configured to ignore GSH TLVs in general, or to ignore GSH TLVs for certain sources or groups. In particular, an administrator might configure a router to not process GSH TLVs if the router is known to never have any directly connected receivers.

For each group that has directly connected receivers, this router SHOULD send PIM (S,G) joins for all the SG mappings advertised in the message for the group. Generally joins are sent, but there could for instance be administrative policy limiting which sources and groups to join. The SG mappings are kept alive for as long as the holdtimer for the source is running. Once the holdtimer expires a PIM router MAY send a PIM (S,G) prune to remove itself from the tree. However, when this happens, there should be no more packets sent by the source, so it may be desirable to allow the state to time out rather than sending a prune.

Note that a holdtime of zero has a special meaning. It is to be treated as if the source just expired, and state to be removed. Source information MUST NOT be removed due to the source being omitted in a message. For instance, if there is a large number of sources for a group, there may be multiple PFM messages, each message containing a different list of sources for the group.

4.4. The First Packets and Bursty Sources

The PIM register procedure is designed to deliver Multicast packets to the RP in the absence of a Shortest Path Tree (SPT) from the RP to the source. The register packets received on the RP are decapsulated and forwarded down the shared tree to the LHRs. As soon as an SPT is built, multicast packets would flow natively over the SPT to the RP or LHR and the register process would stop. The PIM register process ensures packet delivery until an SPT is in place reaching the FHR. If the packets were not unicast encapsulated to the RP they would be dropped by the FHR until the SPT is setup. This functionality is important for applications where the initial packet(s) must be received for the application to work correctly. Another reason would be for bursty sources. If the application sends out a multicast packet every 4 minutes (or longer), the SPT is torn down (typically after 3:30 minutes of inactivity) before the next packet is forwarded down the tree. This will cause no multicast packet to ever be forwarded. A well behaved application should be able to deal with

packet loss since IP is a best effort based packet delivery system. But in reality this is not always the case.

With the procedures defined in this document the packet(s) received by the FHR will be dropped until the LHR has learned about the source and the SPT is built. That means for bursty sources or applications sensitive for the delivery of the first packet this solution would not be very applicable. This solution is mostly useful for applications that don't have strong dependency on the initial packet(s) and have a fairly constant data rate, like video distribution for example. For applications with strong dependency on the initial packet(s) using PIM Bidir [RFC5015] or SSM [RFC4607] is recommended. The protocol operations are much simpler compared to PIM SM, it will cause less churn in the network and both guarantee best effort delivery for the initial packet(s).

4.5. Resiliency to Network Partitioning

In a PIM SM deployment where the network becomes partitioned, due to link or node failure, it is possible that the RP becomes unreachable to a certain part of the network. New sources that become active in that partition will not be able to register to the RP and receivers within that partition are not able to receive the traffic. Ideally you would want to have a candidate RP in each partition, but you never know in advance which routers will form a partitioned network. In order to be fully resilient, each router in the network may end up being a candidate RP. This would increase the operational complexity of the network.

The solution described in this document does not suffer from that problem. If a network becomes partitioned and new sources become active, the receivers in that partitioned will receive the SG Mappings and join the source tree. Each partition works independently of the other partition(s) and will continue to have access to sources within that partition. Once the network has healed, the periodic flooding of SG Mappings ensures that they are re-flooded into the other partition(s) and other receivers can join to the newly learned sources.

5. Configurable Parameters

This document contains a number of configurable parameters. These parameters are formally defined in Section 3.3 and Section 4.2, but they are repeated here for ease of reference. These parameters all have default values as noted below.

Max_PFM_Message_Rate: The maximum number of PFM messages a router is allowed to originate per minute, see Section 3.3 for details. The default value is 6.

Min_PFM_Message_Gap: The minimum amount of time between each PFM message originated by a router in milliseconds, see Section 3.3 for details. The default is 1000.

Group_Source_Holdtime_Period: The announcement period for Group Source Holdtime TLVs in seconds, see Section 4.2 for details. The default value is 60.

Group_Source_Holdtime_Holdtime: The holdtime for Group Source Holdtime TLVs in seconds, see Section 4.2 for details. The default value is 210.

6. Security Considerations

When it comes to general PIM message security, see [RFC7761]. PFM messages **MUST** only be accepted from a PIM neighbor, but as discussed in [RFC7761], any router can become a PIM neighbor by sending a Hello message. To control from where to accept PFM packets, one can limit which interfaces PIM is enabled, and also one can configure interfaces as administrative boundaries for PFM messages, see Section 3.2. The implications of forged PFM messages depend on which TLVs they contain. Documents defining new TLVs will need to discuss the security considerations for the specific TLVs. In general though, the PFM messages are flooded within the network, and by forging a large number of PFM messages one might stress all the routers in the network.

If an attacker can forge PFM messages, then such messages may contain arbitrary GSH TLVs. An issue here is that an attacker might send such TLVs for a huge amount of sources, potentially causing every router in the network to store huge amounts of source state. Also, if there is receiver interest for the groups specified in the GSH TLVs, routers with directly connected receivers will build Shortest Path Trees for the announced sources, even if the sources are not actually active. Building such trees will consume additional resources on routers that the trees pass through.

PIM-SM link-local messages can be authenticated using IPsec, see [RFC7761] section 6.3 and [RFC5796]. Since PFM messages are link-local messages sent hop by hop, a link-local PFM message can be authenticated using IPsec such that a router can verify that a message was sent by a trusted neighbor and has not been modified. However, to verify that a received message contains correct information announced by the originator specified in the message, one

will have to trust every router on the path from the originator and that each router has authenticated the received message.

7. IANA Considerations

This document requires the assignment of a new PIM message type for the PIM Flooding Mechanism (PFM) with the name "PIM Flooding Mechanism". IANA is also requested to create a registry for PFM TLVs called "PIM Flooding Mechanism Message Types". Assignments for the registry are to be made according to the policy "IETF Review" as defined in [RFC8126]. The initial content of the registry should be:

Type	Name	Reference
0	Reserved	[this document]
1	Source Group Holdtime	[this document]
2-32767	Unassigned	

8. Acknowledgments

The authors would like to thank Arjen Boers for contributing to the initial idea, and David Black, Stewart Bryant, Yiqun Cai, Papadimitriou Dimitri, Toerless Eckert, Dino Farinacci, Alvaro Retana and Liang Xia for their very helpful comments on the draft.

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>>.
- [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>>.
- [RFC5796] Atwood, W., Islam, S., and M. Siami, "Authentication and Confidentiality in Protocol Independent Multicast Sparse Mode (PIM-SM) Link-Local Messages", RFC 5796, DOI 10.17487/RFC5796, March 2010, <<https://www.rfc-editor.org/info/rfc5796>>.

- [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>>.

9.2. Informative References

- [RFC3973] Adams, A., Nicholas, J., and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", RFC 3973, DOI 10.17487/RFC3973, January 2005, <<https://www.rfc-editor.org/info/rfc3973>>.
- [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>>.
- [RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <<https://www.rfc-editor.org/info/rfc5015>>.

Authors' Addresses

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

Email: ice@cisco.com

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

Email: stig@cisco.com

Michael Brig
Aegis BMD Program Office
17211 Avenue D, Suite 160
Dahlgren VA 22448-5148
USA

Email: michael.brig@mda.mil

Anders Jonasson
Swedish Defence Material Administration (FMV)
Loennvaegen 4
Vaexjoe 35243
Sweden

Email: anders@jomac.se

PIM Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 13, 2016

X. Liu
Ericsson
P. McAllister
Metaswitch Networks
A. Peter
Juniper Networks
February 10, 2016

A YANG data model for Protocol-Independent Multicast (PIM)
draft-ietf-pim-yang-00

Abstract

This document defines a YANG data model that can be used to configure Protocol Independent Multicast (PIM) 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). 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 August 13, 2016.

Copyright Notice

Copyright (c) 2016 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. 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. Top-level structure	4
2.4. Position of address family in hierarchy	5
3. Unresolved Issues	5
3.1. Current status of work in progress	5
3.2. Group range mappings	5
3.3. Issues blocked on other model designers	6
4. Module Structure	6
4.1. PIM base module	6
4.2. PIM RP module	10
4.3. PIM-SM module	13
4.4. PIM-DM module	14
4.5. PIM-BIDIR module	15
5. PIM YANG Modules	16
5.1. PIM base module	16
5.2. PIM RP module	36
5.3. PIM-SM module	50
5.4. PIM-DM module	57
5.5. PIM-BIDIR module	59
6. TODO list	67
7. Security Considerations	68
8. IANA Considerations	68
9. Acknowledgements	68
10. References	68
10.1. Normative References	68
10.2. Informative References	68
Authors' Addresses	69

1. Introduction

YANG [RFC6020] [RFC6087] 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 draft YANG data model that can be used to configure and manage Protocol-Independent Multicast (PIM) devices. Currently this model is incomplete, but it will support the core PIM protocol, as well as many other features mentioned in separate PIM 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 BCP 14, RFC 2119 [RFC2119].

1.2. Terminology

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

This draft 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 PIM Sparse Mode [RFC4601], including the Source-Specific subset [RFC3569], Dense Mode [RFC3973], and Bi-directional PIM [RFC5015].

The PIM extensions represented in the model include BSR [RFC5059] and Anycast RP [RFC4610].

The representation of some of these features is not completely specified in this draft of the data model. This model is being circulated in its current form for early oversight and review of the basic hierarchy.

The operational state fields and notifications of this model are also incomplete, though the structure of what has been written may be taken as representative of the structure of the model when complete.

This model does not cover other multicast protocols such as IGMP/MLD, MSDP, mVPN, or m-LDP in-band signalling. It does not cover any configuration required to generate the MRIB. These will be covered by future Internet Drafts.

2.2. Optional capabilities

This model is designed to represent the capabilities of PIM devices with various specifications, including some with basic subsets of the PIM protocol. The main design goals of this draft are that any major now-existing implementation may be said to support the base model, and that the configuration of all implementations meeting the

specification is easy to express through some combination of the features in the base 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 PIM implementation.

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 maxima and minima) 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. Top-level structure

This model defines several separate modules for modelling PIM configuration, defined below. Again, this separation will make it easier to express the specific capabilities of a PIM device.

The hierarchy of PIM configuration is designed so that objects that are only relevant for one situation or feature are collected in a container for that feature. For example, the configuration for PIM-SM that is not relevant for an SSM-only implementation is collected in an ASM container.

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 they need not be explicitly configured.

This module structure also applies, where applicable, to the operational state and notifications of the model.

2.4. Position of address family in hierarchy

The current draft contains address-family as a node in the hierarchy multiple times: both under the interface list, and under the PIM instance. This is similar to the IS-IS yang draft model.

The reasoning for this is to make it easier for implementations in which configuration options are not supported for specific address families.

For these implementations, the restriction that interface configuration must be address-family independent must either be expressed as a vendor augmentation of an address-family-independent parameter above the address-family level, or by a constraint on the base model objects of a form similar to:

```
must ". = ../../address-family[address-family='ipv4']/  
interface[interface=current()/sibling:interface]/dr-priority" {  
  error-app-tag dr-priority-mismatch; error-message "Error: IPv6 DR  
  priority must match IPv4 DR priority "; }
```

3. Unresolved Issues

3.1. Current status of work in progress

The model so far details how the PIM modules interact and covers the higher levels of their hierarchy. Some details of interface configuration, RP configuration, and PIM-ASM-specific parameters are also complete.

For a list of the most substantial areas still to cover, please see the "TODO list" section below.

3.2. Group range mappings

There is currently no convenient way in the operational state model to map from a group address to the PIM mode and RP information that it will be forwarded according to, which complicates reasoning about the running state and the diagnosis of conflicting policy configuration.

A hypothetical group range state reporting object indexed by group range would be desirable for this purpose, but would be inconvenient for many implementations that index the relevant information on RP address (not applicable for DM or unroutable group-ranges), and difficult to express directly in YANG (as it is difficult to express a container indexed on an arbitrary, proprietary policy structure).

3.3. Issues blocked on other model designers

Some questions must be resolved with reference to other yang models, and so the resolution may be blocked on a decision from another body. Currently these issues are:

1. The position of BFD in the configuration; does the BFD model augment various different protocols, or do the protocols have to augment themselves individually to support BFD?
2. The abstract concepts of a "set of IP addresses" or "set of IP prefixes", as represented in implementations by prefixes, ACLs or policy statements, is a general concept out of the scope of this document. It will presumably have a yang data-type the instantiation of which is vendor-specific.

4. Module Structure

4.1. PIM base module

The PIM base module defines the router-wide configuration options not specific to any PIM mode, and is included by the other modules. There are a couple of things worth mentioning here regarding where the PIM model fits in the overall routing hierarchy:

1. Our current direction is to agree to a routing-instance-centric (VRF) model as opposed to protocol-centric mainly because it fits well into the routing-instance model, and it is easier to map from the VRF-centric to the protocol-centric than the other way around due to forward references.
2. The PIM base model will augment `/rt:routing/rt:routing-instance/rt:routing-protocols:` as opposed to augmenting `/rt:routing/rt:routing-instance/rt:routing-protocols/rt:routing-protocol` as the latter would allow multiple protocol instances per VRF, which does not make sense for PIM.

```

    module: ietf-pim-base
augment /rt:routing/rt:routing-instance/rt:routing-protocols:
  +--rw pim
    +--rw graceful-restart
      |   +--rw enabled?      boolean
      |   +--rw duration?    uint16
    +--rw address-family* [address-family]
      |   +--rw address-family      identityref
      |   +--rw graceful-restart
          +--rw enabled?      boolean
          +--rw duration?    uint16

```

```

+--rw interfaces
  +--rw interface* [interface]
    +--rw interface          if:interface-ref
    +--rw address-family* [address-family]
      +--rw address-family    identityref
      +--rw bfd
        | +--rw enabled?                boolean
        | +--rw local-multiplier?        multiplier
        | +--rw (interval-config-type)?
        |   +--:(tx-rx-intervals)
        |     | +--rw desired-min-tx-interval    uint32
        |     | +--rw required-min-rx-interval    uint32
        |     +--:(single-interval)
        |       +--rw min-interval                uint32
        +--rw dr-priority?          uint32 {_intf-dr-priority}?
        +--rw hello-interval?        timer-value {_intf-hello-interval}?
        +--rw (hello-holdtime-or-multiplier)?
        |   +--:(holdtime) {_intf-hello-holdtime}?
        |     | +--rw hello-holdtime?          timer-value
        |     +--:(multiplier) {_intf-hello-multiplier}?
        |       +--rw hello-multiplier?          uint8
        +--rw jp-interval?          timer-value {_intf-jp-interval}?
        +--rw (jp-holdtime-or-multiplier)?
        |   +--:(holdtime) {_intf-jp-holdtime}?
        |     | +--rw jp-holdtime?              timer-value
        |     +--:(multiplier) {_intf-jp-multiplier}?
        |       +--rw jp-multiplier?            uint8
        +--rw propagation-delay?    uint16 {_intf-propagation-delay}?
        +--rw override-interval?    uint16 {_intf-override-interval}?
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols:
+--ro pim
  +--ro address-family* [address-family]
    | +--ro address-family    identityref
    | +--ro statistics
    |   +--ro discontinuity-time?  yang:date-and-time
    |   +--ro error
    |     | +--ro assert?                yang:counter64
    |     | +--ro bsr?                  yang:counter64
    |     | +--ro candidate-rp-advertisement?  yang:counter64
    |     | +--ro hello?                yang:counter64
    |     | +--ro join-prune?           yang:counter64
    |     | +--ro register?             yang:counter64
    |     | +--ro register-stop?        yang:counter64
    |     | +--ro state-refresh?        yang:counter64
    |   +--ro queue
    |     | +--ro size?                uint32
    |     | +--ro overflow?            yang:counter32
    |   +--ro received

```

```

|--ro assert? yang:counter64
|--ro bsr? yang:counter64
|--ro candidate-rp-advertisement? yang:counter64
|--ro hello? yang:counter64
|--ro join-prune? yang:counter64
|--ro register? yang:counter64
|--ro register-stop? yang:counter64
|--ro state-refresh? yang:counter64
+--ro sent
  |--ro assert? yang:counter64
  |--ro bsr? yang:counter64
  |--ro candidate-rp-advertisement? yang:counter64
  |--ro hello? yang:counter64
  |--ro join-prune? yang:counter64
  |--ro register? yang:counter64
  |--ro register-stop? yang:counter64
  |--ro state-refresh? yang:counter64
+--ro topology-tree-info
  +--ro ipv4-route* [group source-addr is-rpt]
    |--ro group inet:ipv4-address
    |--ro source-addr union
    |--ro is-rpt boolean
    |--ro expire? uint32
    |--ro incoming-interface? if:interface-ref
    |--ro mode? pim-mode
    |--ro msdp-learned? boolean
    |--ro rp-address? inet:ip-address
    |--ro rpf-neighbor? inet:ip-address
    |--ro spt-bit? boolean
    |--ro up-time? uint32
    +--ro outgoing-interface* [name]
      |--ro name if:interface-ref
      |--ro expire? timer-value
      |--ro up-time? uint32
      |--ro jp-state? enumeration
  +--ro ipv6-route* [group source-addr is-rpt]
    |--ro group inet:ipv6-address
    |--ro source-addr union
    |--ro is-rpt boolean
    |--ro expire? uint32
    |--ro incoming-interface? if:interface-ref
    |--ro mode? pim-mode
    |--ro msdp-learned? boolean
    |--ro rp-address? inet:ip-address
    |--ro rpf-neighbor? inet:ip-address
    |--ro spt-bit? boolean
    |--ro up-time? uint32
    +--ro outgoing-interface* [name]

```

```

|         +--ro name          if:interface-ref
|         +--ro expire?       timer-value
|         +--ro up-time?      uint32
|         +--ro jp-state?     enumeration
+--ro interfaces
  +--ro interface* [interface]
    +--ro interface          if:interface-ref
    +--ro address-family* [address-family]
      +--ro address-family    identityref
      +--ro bfd
        +--ro enabled?                boolean
        +--ro local-multiplier?       multiplier
        +--ro (interval-config-type)?
          +--:(tx-rx-intervals)
            +--ro desired-min-tx-interval    uint32
            +--ro required-min-rx-interval    uint32
          +--:(single-interval)
            +--ro min-interval                uint32
      +--ro dr-priority?      uint32 {intf-dr-priority}?
      +--ro hello-interval?   timer-value {intf-hello-interval}?
      +--ro (hello-holdtime-or-multiplier)?
        +--:(holdtime) {intf-hello-holdtime}?
        | +--ro hello-holdtime?   timer-value
        +--:(multiplier) {intf-hello-multiplier}?
        | +--ro hello-multiplier?  uint8
      +--ro jp-interval?      timer-value {intf-jp-interval}?
      +--ro (jp-holdtime-or-multiplier)?
        +--:(holdtime) {intf-jp-holdtime}?
        | +--ro jp-holdtime?      timer-value
        +--:(multiplier) {intf-jp-multiplier}?
        | +--ro jp-multiplier?    uint8
      +--ro propagation-delay? uint16 {intf-propagation-delay}?
      +--ro override-interval? uint16 {intf-override-interval}?
      +--ro ipv4
        +--ro address*        inet:ipv4-address
        +--ro dr-addr?        inet:ipv4-address
      +--ro ipv6
        +--ro address*        inet:ipv6-address
        +--ro dr-addr?        inet:ipv6-address
      +--ro oper-status?      enumeration
      +--ro hello-expire?     timer-value
      +--ro neighbor-ipv4* [address]
        +--ro address          inet:ipv4-address
        +--ro bfd-status?      enumeration
        +--ro expire?          timer-value
        +--ro dr-priority?     uint32
        +--ro gen-id?          uint32
        +--ro up-time?         uint32

```

```

        +--ro neighbor-ipv6* [address]
            +--ro address          inet:ipv6-address
            +--ro bfd-status?      enumeration
            +--ro expire?          timer-value
            +--ro dr-priority?     uint32
            +--ro gen-id?          uint32
            +--ro up-time?         uint32
notifications:
+---n pim-neighbor-event
|   +--ro event-type?            neighbor-event-type
|   +--ro routing-instance-state-ref? rt:routing-instance-state-ref
|   +--ro interface-state-ref?     leafref
|   +--ro interface-af-state-ref?  leafref
|   +--ro neighbor-ipv4-state-ref? leafref
|   +--ro neighbor-ipv6-state-ref? leafref
|   +--ro up-time?                uint32
+---n pim-interface-event
|   +--ro event-type?            interface-event-type
|   +--ro routing-instance-state-ref? rt:routing-instance-state-ref
|   +--ro interface-state-ref?     leafref
|   +--ro ipv4
|   |   +--ro address*          inet:ipv4-address
|   |   +--ro dr-addr?          inet:ipv4-address
|   +--ro ipv6
|   |   +--ro address*          inet:ipv6-address
|   |   +--ro dr-addr?          inet:ipv6-address

```

4.2. PIM RP module

The PIM RP module contains configuration information scoped to RPs or ranges of group addresses. This does not belong in the hierarchy under any PIM mode, but is augmented by the individual mode-specific modules as appropriate.

```

module: ietf-pim-rp
augment /rt:routing/rt:routing-instance/rt:routing-protocols/pim-base:pim/
pim-base:address-family:
+--rw rp
+--rw static-rp
|   +--rw ipv4-rp* [ipv4-addr]
|   |   +--rw ipv4-addr    inet:ipv4-address
|   +--rw ipv6-rp* [ipv6-addr]
|   |   +--rw ipv6-addr    inet:ipv6-address
+--rw bsr {bsr}?
|   +--rw bsr-candidate!
|   |   +--rw (interface-or-address)?
|   |   |   +--:(interface) {candidate-interface}?

```

```

|   |   |   |--rw interface                if:interface-ref
|   |   |   +---:(ipv4-address) {candidate-ipv4}?
|   |   |   |   |--rw ipv4-address          inet:ipv4-address
|   |   |   +---:(ipv6-address) {candidate-ipv6}?
|   |   |   |   |--rw ipv6-address          inet:ipv6-address
|   |--rw hash-mask-length    uint8
|   |--rw priority            uint8
+---rw rp-candidate-interface* [interface] {candidate-interface}?
|   |--rw interface          if:interface-ref
|   |--rw policy?            string
|   |--rw mode?              identityref
+---rw rp-candidate-ipv4-address* [ipv4-address] {candidate-ipv4}?
|   |--rw ipv4-address        inet:ipv4-address
|   |--rw policy?            string
|   |--rw mode?              identityref
+---rw rp-candidate-ipv6-address* [ipv6-address] {candidate-ipv6}?
|   |--rw ipv6-address        inet:ipv6-address
|   |--rw policy?            string
|   |--rw mode?              identityref
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:
  +--ro rp
    +--ro static-rp
      |--ro ipv4-rp* [ipv4-addr]
      |   |--ro ipv4-addr    inet:ipv4-address
      |--ro ipv6-rp* [ipv6-addr]
      |   |--ro ipv6-addr    inet:ipv6-address
    +--ro bsr {bsr}?
      +--ro bsr-candidate!
        |--ro (interface-or-address)?
        |   +---:(interface) {candidate-interface}?
        |   |   |--ro interface          if:interface-ref
        |   +---:(ipv4-address) {candidate-ipv4}?
        |   |   |--ro ipv4-address        inet:ipv4-address
        |   +---:(ipv6-address) {candidate-ipv6}?
        |   |   |--ro ipv6-address        inet:ipv6-address
        |--ro hash-mask-length    uint8
        |--ro priority            uint8
      +--ro rp-candidate-interface* [interface] {candidate-interface}?
      |   |--ro interface          if:interface-ref
      |   |--ro policy?            string
      |   |--ro mode?              identityref
      +--ro rp-candidate-ipv4-address* [ipv4-address] {candidate-ipv4}?
      |   |--ro ipv4-address        inet:ipv4-address
      |   |--ro policy?            string
      |   |--ro mode?              identityref
      +--ro rp-candidate-ipv6-address* [ipv6-address] {candidate-ipv6}?
      |   |--ro ipv6-address        inet:ipv6-address

```

```

| |   +--ro policy?          string
| |   +--ro mode?           identityref
| +--ro bsr
| |   +--ro addr?           inet:ip-address
| |   +--ro hash-mask-length? uint8
| |   +--ro priority?       uint8
| |   +--ro up-time?        uint32
| +--ro (election-state)? {bsr-election-state}?
| |   +--:(candidate)
| | |   +--ro candidate-bsr-state?          enumeration
| | +--:(non-candidate)
| | |   +--ro non-candidate-bsr-state?      enumeration
| +--ro bsr-next-bootstrap?                uint16
| +--ro rp
| |   +--ro rp-address?      inet:ip-address
| |   +--ro group-policy?    string
| |   +--ro up-time?        uint32
| +--ro rp-candidate-next-advertisement?    uint16
+--ro rp-list
| +--ro ipv4-rp* [ipv4-addr mode]
| |   +--ro ipv4-addr        inet:ipv4-address
| |   +--ro mode             identityref
| |   +--ro info-source-addr? inet:ipv4-address
| |   +--ro info-source-type? identityref
| |   +--ro up-time?         uint32
| |   +--ro expire?          pim-base:timer-value
| +--ro ipv6-rp* [ipv6-addr mode]
| |   +--ro ipv6-addr        inet:ipv6-address
| |   +--ro mode             identityref
| |   +--ro info-source-addr? inet:ipv6-address
| |   +--ro info-source-type? identityref
| |   +--ro up-time?         uint32
| |   +--ro expire?          pim-base:timer-value
+--ro rp-mappings
| +--ro ipv4-rp* [group rp-addr]
| |   +--ro group            inet:ipv4-prefix
| |   +--ro rp-addr          inet:ipv4-address
| |   +--ro up-time?         uint32
| |   +--ro expire?          pim-base:timer-value
| +--ro ipv6-rp* [group rp-addr]
| |   +--ro group            inet:ipv6-prefix
| |   +--ro rp-addr          inet:ipv6-address
| |   +--ro up-time?         uint32
| |   +--ro expire?          pim-base:timer-value
notifications:
+---n pim-rp-event
|   +--ro event-type?          rp-event-type
|   +--ro routing-instance-state-ref? rt:routing-instance-state-ref

```

```

+--ro instance-af-state-ref?      leafref
+--ro group?                      inet:ip-address
+--ro rp-address?                 inet:ip-address
+--ro is-rpt?                     boolean
+--ro mode?                       pim-base:pim-mode
+--ro message-origin?             inet:ip-address

```

4.3. PIM-SM module

This module covers Sparse Mode configuration, including PIM-ASM and PIM-SSM.

```

      module: ietf-pim-sm
augment /rt:routing/rt:routing-instance/rt:routing-protocols/pim-base:pim/
pim-base:address-family:
  +--rw sm
    +--rw asm
      +--rw anycast-rp!
        +--rw ipv4
          +--rw ipv4-anycast-rp* [anycast-addr rp-addr]
            +--rw anycast-addr    inet:ipv4-address
            +--rw rp-addr         inet:ipv4-address
        +--rw ipv6
          +--rw ipv6-anycast-rp* [anycast-addr rp-addr]
            +--rw anycast-addr    inet:ipv6-address
            +--rw rp-addr         inet:ipv6-address
      +--rw spt-switch
        +--rw infinity! {spt-switch-infinity}?
        +--rw policy-name?   string {spt-switch-policy}?
    +--rw ssm!
      +--rw range-policy?    string
augment /rt:routing/rt:routing-instance/rt:routing-protocols/pim-base:pim/
pim-base:interfaces/pim-base:interface/pim-base:address-family:
  +--rw sm!
    +--rw passive?          empty
augment /rt:routing/rt:routing-instance/rt:routing-protocols/pim-base:pim/
pim-base:address-family/pim-rp:rp/pim-rp:static-rp/pim-rp:ipv4-rp:
  +--rw sm!
    +--rw policy-name?      string
    +--rw override?         boolean {static-rp-override}?
augment /rt:routing/rt:routing-instance/rt:routing-protocols/pim-base:pim/
pim-base:address-family/pim-rp:rp/pim-rp:static-rp/pim-rp:ipv6-rp:
  +--rw sm!
    +--rw policy-name?      string
    +--rw override?         boolean {static-rp-override}?
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:

```



```

+--ro sm
  +--ro asm
    |   +--ro anycast-rp!
    |   |   +--ro ipv4
    |   |   |   +--ro ipv4-anycast-rp* [anycast-addr rp-addr]
    |   |   |   |   +--ro anycast-addr      inet:ipv4-address
    |   |   |   |   +--ro rp-addr          inet:ipv4-address
    |   |   +--ro ipv6
    |   |   |   +--ro ipv6-anycast-rp* [anycast-addr rp-addr]
    |   |   |   |   +--ro anycast-addr      inet:ipv6-address
    |   |   |   |   +--ro rp-addr          inet:ipv6-address
    |   +--ro spt-switch
    |   |   +--ro infinity! {spt-switch-infinity}?
    |   |   +--ro policy-name?   string {spt-switch-policy}?
  +--ro ssm!
    +--ro range-policy?   string
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:interfaces/pim-base:interface/
pim-base:address-family:
  +--ro sm!
    +--ro passive?       empty
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp/pim-rp:static-rp/
pim-rp:ipv4-rp:
  +--ro sm!
    +--ro policy-name?   string
    +--ro override?      boolean {static-rp-override}?
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp/pim-rp:static-rp/
pim-rp:ipv6-rp:
  +--ro sm!
    +--ro policy-name?   string
    +--ro override?      boolean {static-rp-override}?

```

4.4. PIM-DM module

This module will cover Dense Mode configuration.

```

module: ietf-pim-dm
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:
  +--rw dm!
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:interfaces/pim-base:interface/
pim-base:address-family:
  +--rw dm!
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:
  +--ro dm

```

4.5. PIM-BIDIR module

This module will cover Bidirectional PIM configuration.

```

module: ietf-pim-bidir
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:
  +--rw bidir
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:interfaces/pim-base:interface/
pim-base:address-family:
  +--rw bidir!
    +--rw df-election {intf-df-election}?
      +--rw offer-interval?      pim-base:timer-value
      +--rw backoff-interval?    pim-base:timer-value
      +--rw offer-multiplier?    uint8
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp/pim-rp:static-rp/
pim-rp:ipv4-rp:
  +--rw bidir!
    +--rw policy-name?    string
    +--rw override?       boolean {static-rp-override}?
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp/pim-rp:static-rp/
pim-rp:ipv6-rp:
  +--rw bidir!
    +--rw policy-name?    string
    +--rw override?       boolean {static-rp-override}?
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:
  +--ro bidir
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:interfaces/pim-base:interface/
pim-base:address-family:

```

```

    +--ro bidir!
      +--ro df-election {intf-df-election}?
        +--ro offer-interval?      pim-base:timer-value
        +--ro backoff-interval?    pim-base:timer-value
        +--ro offer-multiplier?    uint8
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp/pim-rp:static-rp/
pim-rp:ipv4-rp:
  +--ro bidir!
    +--ro policy-name?      string
    +--ro override?         boolean {static-rp-override}?
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp/pim-rp:static-rp/
pim-rp:ipv6-rp:
  +--ro bidir!
    +--ro policy-name?      string
    +--ro override?         boolean {static-rp-override}?
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp:
  +--ro bidir
    +--ro df-election
      | +--ro ipv4-rp* [ipv4-addr]
      | | +--ro ipv4-addr      inet:ipv4-address
      | +--ro ipv6-rp* [ipv6-addr]
      | | +--ro ipv6-addr      inet:ipv6-address
    +--ro interface-df-election
      +--ro ipv4-rp* [ipv4-addr interface-name]
      | +--ro ipv4-addr          inet:ipv4-address
      | +--ro interface-name     if:interface-ref
      | +--ro df-address?        inet:ipv4-address
      | +--ro interface-state?   identityref
      +--ro ipv6-rp* [ipv6-addr interface-name]
      | +--ro ipv6-addr          inet:ipv6-address
      | +--ro interface-name     if:interface-ref
      | +--ro df-address?        inet:ipv6-address
      | +--ro interface-state?   identityref

```

5. PIM YANG Modules

5.1. PIM base module

```

<CODE BEGINS> file "ietf-pim-base.yang"

module ietf-pim-base {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pim-base";
  // replace with IANA namespace when assigned

```

```
prefix pim-base;

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

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

import ietf-interfaces {
  prefix "if";
}

import ietf-routing {
  prefix "rt";
}

import ietf-bfd {
  prefix "bfd";
}

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:   ";

description
  "The module defines a collection of YANG definitions common for
  all PIM modes.";

revision 2015-12-07 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for PIM";
}
```

```
/*
 * Features
 */
feature bfd-protocol-parms {
  description
    "BFD protocol specific parameters support.";
}

feature global-graceful-restart {
  description
    "Global configuraiont for graceful restart support as per
    RFC5306.";
}

feature intf-dr-priority {
  description
    "Support configuration of interface dr priority.";
}

feature intf-hello-holdtime {
  description
    "Support configuration of interface hello holdtime.";
}

feature intf-hello-interval {
  description
    "Support configuration of interface hello interval.";
}

feature intf-hello-multiplier {
  description
    "Support configuration of interface hello multiplier.";
}

feature intf-jp-interval {
  description
    "Support configuration of interface join prune interval.";
}

feature intf-jp-holdtime {
  description
    "Support configuration of interface join prune holdtime.";
}

feature intf-jp-multiplier {
  description
    "Support configuration of interface join prune multiplier.";
}
```

```
feature intf-propagation-delay {
  description
    "Support configuration of interface propagation delay.";
}

feature intf-override-interval {
  description
    "Support configuration of interface override interval.";
}

feature per-af-graceful-restart {
  description
    "Per AF configuraiont for graceful restart support as per
    RFC5306.";
}

/*
 * Typedefs
 */
typedef interface-event-type {
  type enumeration {
    enum up {
      description
        "Neighbor status changed to up.";
    }
    enum down {
      description
        "Neighbor status changed to down.";
    }
    enum new-dr {
      description
        "A new DR was elected on the connected network.";
    }
    enum new-df {
      description
        "A new DF was elected on the connected network.";
    }
  }
  description "Operational status event type for notifications.";
}

typedef neighbor-event-type {
  type enumeration {
    enum up {
      description
        "Neighbor status changed to up.";
    }
    enum down {
```

```
        description
            "Neighbor status changed to down.";
    }
}
description "Operational status event type for notifications.";
}

typedef pim-mode {
    type enumeration {
        enum none {
            description
                "PIM is not operating.";
        }
        enum ssm {
            description
                "Source-Specific Multicast (SSM) with PIM Sparse Mode.";
        }
        enum asm {
            description
                "Any Source Multicast (ASM) with PIM Sparse Mode.";
        }
        enum bidir {
            description
                "Bidirectional PIM.";
        }
        enum dm {
            description
                "PIM Dense Mode.";
        }
        enum other {
            description
                "Any other PIM mode.";
        }
    }
}
description
    "The PIM mode in which a group is operating.";
}

typedef timer-value {
    type union {
        type uint16;
        type enumeration {
            enum "infinity" {
                description "The timer is set to infinity.";
            }
            enum "no-expiry" {
                description "The timer is not set.";
            }
        }
    }
}
```

```
    }
  }
  units seconds;
  description "Timer value type.";
} // timer-value

/*
 * Identities
 */

/*
 * Groupings
 */
grouping global-attributes {
  description
    "A Grouping defining global configuration attributes.";
  uses graceful-restart-container {
    if-feature global-graceful-restart;
  }
} // global-attributes

grouping graceful-restart-container {
  description
    "A grouping defining a container of graceful restart
    attributes.";
  container graceful-restart {
    leaf enabled {
      type boolean;
      description
        "Enable or disable graceful restart.";
    }
    leaf duration {
      type uint16;
      units seconds;
      description
        "Maximum time for graceful restart to finish.";
    }
  }
  description
    "Container of graceful restart attributes.";
}
} // graceful-restart-container

grouping interface-config-attributes {
  description
    "A grouping defining interface attributes.";
  container bfd {
    description "BFD operation.";
    leaf enabled {
```



```
        type boolean;
        description
            "True if BFD is enabled for the interface.";
    }
    uses bfd:bfd-grouping-base-cfg-parms {
        if-feature bfd-protocol-parms;
    }
}
leaf dr-priority {
    if-feature intf-dr-priority;
    type uint32;
    description "DR priority";
}
leaf hello-interval {
    if-feature intf-hello-interval;
    type timer-value;
    description "Hello interval";
}
choice hello-holdtime-or-multiplier {
    description "Use holdtime or multiplier";
    case holdtime {
        if-feature intf-hello-holdtime;
        leaf hello-holdtime {
            type timer-value;
            description "Hello holdtime";
        }
    }
    case multiplier {
        if-feature intf-hello-multiplier;
        leaf hello-multiplier {
            type uint8;
            description "Hello multiplier";
        }
    }
}
leaf jp-interval {
    if-feature intf-jp-interval;
    type timer-value;
    description "Join prune interval";
}
choice jp-holdtime-or-multiplier {
    description "Use holdtime or multiplier";
    case holdtime {
        if-feature intf-jp-holdtime;
        leaf jp-holdtime {
            type timer-value;
            description "Join prune holdtime";
        }
    }
}
```

```
    }
    case multipler {
      if-feature intf-jp-multipler;
      leaf jp-multipler {
        type uint8;
        description "Join prune multipler";
      }
    }
  }
  leaf propagation-delay {
    if-feature intf-propagation-delay;
    type uint16;
    units milliseconds;
    description "Propagation description";
  }
  leaf override-interval {
    if-feature intf-override-interval;
    type uint16;
    units milliseconds;
    description "Override interval";
  }
} // interface-config-attributes

grouping interface-state-attributes {
  description
    "A grouping defining interface attributes.";
  container ipv4 {
    when "../../../address-family = 'rt:ipv4'" {
      description
        "Only applicable to ipv4 address family.";
    }
    description "";
    leaf-list address {
      type inet:ipv4-address;
      description "";
    }
    leaf dr-addr {
      type inet:ipv4-address;
      description "";
    }
  }
  container ipv6 {
    when "../../../address-family = 'rt:ipv6'" {
      description
        "Only applicable to ipv6 address family.";
    }
    description "";
    leaf-list address {
```

```
        type inet:ipv6-address;
        description "";
    }
    leaf dr-addr {
        type inet:ipv6-address;
        description "";
    }
}
uses interface-state-af-attributes;
} // interface-state-attributes

grouping interface-state-af-attributes {
    description
        "A grouping defining interface per af attributes.";

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

    leaf hello-expire {
        type timer-value;
        description "Hello interval exiration time.";
    }

    list neighbor-ipv4 {
        when "../..../address-family = 'rt:ipv4'" {
            description
                "Only applicable to ipv4 address family.";
        }
        key "address";
        description "";
        leaf address {
            type inet:ipv4-address;
            description "";
        }
        uses neighbor-state-af-attributes;
    } // list neighbor-ipv4
}
```

```
list neighbor-ipv6 {
  when "../.../address-family = 'rt:ipv6'" {
    description
      "Only applicable to ipv6 address family.";
  }
  key "address";
  description "";
  leaf address {
    type inet:ipv6-address;
    description "";
  }
  uses neighbor-state-af-attributes;
} // list neighbor-ipv6
} // interface-state-af-attributes

grouping multicast-route-attributes {
  description
    "A grouping defining multicast route attributes.";

  leaf expire {
    type uint32;
    units seconds;
    description "";
  }
  leaf incoming-interface {
    type if:interface-ref;
    description
      "Reference to an entry in the global interface
      list.";
  }
  leaf mode {
    type pim-mode;
    description "";
  }
  leaf msdp-learned {
    type boolean;
    description "";
  }
  leaf rp-address {
    type inet:ip-address;
    description "";
  }
  leaf rpf-neighbor {
    type inet:ip-address;
    description "";
  }
  leaf spt-bit {
    type boolean;
```

```
        description "";
    }
    leaf up-time {
        type uint32;
        units seconds;
        description "";
    }
    list outgoing-interface {
        key "name";
        description
            "A list of outgoing interfaces.";

        leaf name {
            type if:interface-ref;
            description
                "Interface name";
        }

        leaf expire {
            type timer-value;
            description "Expiring information.";
        }

        leaf up-time {
            type uint32;
            units seconds;
            description "";
        }

        leaf jp-state {
            type enumeration {
                enum "no-info" {
                    description
                        "The interface has Join state and no timers running";
                }
                enum "join" {
                    description
                        "The interface has Join state.";
                }
                enum "prune-pending" {
                    description
                        "The router has received a Prune on this interface from
                        a downstream neighbor and is waiting to see whether
                        the prune will be overridden by another downstream
                        router. For forwarding purposes, the Prune-Pending
                        state functions exactly like the Join state.";
                }
            }
        }
    }
}
```

```
        description "";
    }
} // multicast-route-attributes

grouping neighbor-state-af-attributes {
    description
        "A grouping defining neighbor per af attributes.";
    leaf bfd-status {
        type enumeration {
            enum up {
                description
                    "";
            }
            enum down {
                description
                    "";
            }
        }
        description "";
    }
    leaf expire {
        type timer-value;
        description "Neighbor expiring information.";
    }
    leaf dr-priority {
        type uint32;
        description "DR priority";
    }
    leaf gen-id {
        type uint32;
        description "Generation ID.";
    }
    leaf up-time {
        type uint32;
        units seconds;
        description "";
    }
} // neighbor-state-af-attributes

grouping per-af-attributes {
    description
        "A grouping defining per address family attributes.";
    uses graceful-restart-container {
        if-feature per-af-graceful-restart;
    }
} // per-af-attributes
```

```
grouping pim-instance-state-ref {
  description
    "An absolute reference to a PIM instance.";
  leaf routing-instance-state-ref {
    type rt:routing-instance-state-ref;
    description
      "Reference to the routing instance state.";
  }
} // pim-instance-state-ref

grouping pim-instance-af-state-ref {
  description
    "An absolute reference to a PIM instance address family.";
  uses pim-instance-state-ref;
  leaf instance-af-state-ref {
    type leafref {
      path "/rt:routing-state/rt:routing-instance"
        + "[rt:name = current()../routing-instance-state-ref]/"
        + "rt:routing-protocols/pim-base:pim/"
        + "pim-base:address-family/pim-base:address-family";
    }
    description
      "Reference to a PIM instance address family.";
  }
} // pim-instance-state-af-ref

grouping pim-interface-state-ref {
  description
    "An absolute reference to a PIM interface state.";
  uses pim-instance-state-ref;
  leaf interface-state-ref {
    type leafref {
      path "/rt:routing-state/rt:routing-instance"
        + "[rt:name = current()../routing-instance-state-ref]/"
        + "rt:routing-protocols/pim-base:pim/pim-base:interfaces/"
        + "pim-base:interface/pim-base:interface";
    }
    description
      "Reference to a PIM interface.";
  }
} // pim-interface-state-ref

grouping pim-neighbor-state-ref {
  description
    "An absolute reference to a PIM neighbor state.";
  uses pim-interface-state-ref;
  leaf interface-af-state-ref {
    type leafref {
```

```
    path "/rt:routing-state/rt:routing-instance"
      + "[rt:name = current()/../routing-instance-state-ref]/"
      + "rt:routing-protocols/pim-base:pim/pim-base:interfaces/"
      + "pim-base:interface"
      + "[pim-base:interface = "
      + "current()/../interface-state-ref]/"
      + "pim-base:address-family/pim-base:address-family";
  }
  description
    "Reference to a PIM interface address family.";
}
leaf neighbor-ipv4-state-ref {
  when "../interface-af-state-ref = 'rt:ipv4'" {
    description "";
  }
  type leafref {
    path "/rt:routing-state/rt:routing-instance"
      + "[rt:name = current()/../routing-instance-state-ref]/"
      + "rt:routing-protocols/pim-base:pim/pim-base:interfaces/"
      + "pim-base:interface"
      + "[pim-base:interface = "
      + "current()/../interface-state-ref]/"
      + "pim-base:address-family"
      + "[pim-base:address-family = "
      + "current()/../interface-af-state-ref]/"
      + "pim-base:neighbor-ipv4/pim-base:address";
  }
  description
    "Reference to a PIM IPv4 neighbor.";
}
leaf neighbor-ipv6-state-ref {
  when "../interface-af-state-ref = 'rt:ipv6'" {
    description "";
  }
  type leafref {
    path "/rt:routing-state/rt:routing-instance"
      + "[rt:name = current()/../routing-instance-state-ref]/"
      + "rt:routing-protocols/pim-base:pim/pim-base:interfaces/"
      + "pim-base:interface"
      + "[pim-base:interface = "
      + "current()/../interface-state-ref]/"
      + "pim-base:address-family"
      + "[pim-base:address-family = "
      + "current()/../interface-af-state-ref]/"
      + "pim-base:neighbor-ipv6/pim-base:address";
  }
  description
    "Reference to a PIM IPv6 neighbor.";
```



```
    }  
  } // pim-neighbor-state-ref  
  
  grouping statistics-container {  
    description  
      "A container defining statistics attributes.";  
    container statistics {  
      description "";  
      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 "";  
        uses statistics-error;  
      }  
      container queue {  
        description "";  
        uses statistics-queue;  
      }  
      container received {  
        description "";  
        uses statistics-sent-received;  
      }  
      container sent {  
        description "";  
        uses statistics-sent-received;  
      }  
    }  
  }  
} // statistics-container  
  
grouping statistics-error {  
  description  
    "A grouping defining error statistics  
    attributes.";  
  uses statistics-sent-received;  
}  
// statistics-error  
  
grouping statistics-queue {  
  description  
    "A grouping defining queue statistics  
    attributes.";
```

```
    leaf size {
      type uint32;
      description
        "The size of the input queue.";
    }
    leaf overflow {
      type yang:counter32;
      description
        "The number of the input queue overflows.";
    }
  } // statistics-queue

grouping statistics-sent-received {
  description
    "A grouping defining sent and received statistics
    attributes.";
  leaf assert {
    type yang:counter64;
    description
      "The number of assert messages.";
  }
  leaf bsr {
    type yang:counter64;
    description
      "The number of bsr messages.";
  }
  leaf candidate-rp-advertisement {
    type yang:counter64;
    description
      "The number of Candidate-RP-advertisement messages.";
  }
  leaf hello {
    type yang:counter64;
    description
      "The number of hello messages.";
  }
  leaf join-prune {
    type yang:counter64;
    description
      "The number of join/prune messages.";
  }
  leaf register {
    type yang:counter64;
    description
      "The number of register messages.";
  }
  leaf register-stop {
    type yang:counter64;
```

```
        description
            "The number of register stop messages.";
    }
    leaf state-refresh {
        type yang:counter64;
        description
            "The number of state refresh messages.";
    }
} // statistics-sent-received

/*
 * Configuration data nodes
 */

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols" {
    description
        "PIM augmentation to routing instance configuration.";

    container pim {
        description
            "PIM configuration data.";

        uses global-attributes;

        list address-family {
            key "address-family";
            description
                "Each list entry for one address family.";
            uses rt:address-family;
            uses per-af-attributes;

        } // address-family

        container interfaces {
            description
                "Containing a list of interfaces.";
            list interface {
                key "interface";
                description
                    "List of pim interfaces.";
                leaf interface {
                    type if:interface-ref;
                    description
                        "Reference to an entry in the global interface
                        list.";
                }
                list address-family {
```

```

        key "address-family";
        description
            "Each list entry for one address family.";
        uses rt:address-family;
        uses interface-config-attributes;
    } // address-family
} // interface
} // interfaces
} // pim
} // augment

/*
 * Operational state data nodes
 */

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols" {
    description
        "PIM augmentation to routing instance state.";
    container pim {
        description
            "PIM state data.";

        list address-family {
            key "address-family";
            description
                "Each list entry for one address family.";
            uses rt:address-family;

            uses statistics-container;

            container topology-tree-info {
                description "";
                list ipv4-route {
                    when "../../address-family = 'rt:ipv4'" {
                        description
                            "Only applicable to ipv4 address family.";
                    }
                    key "group source-addr is-rpt";
                    description "";
                    leaf group {
                        type inet:ipv4-address;
                        description "";
                    }
                    leaf source-addr {
                        type union {
                            type enumeration {
                                enum '*' {

```

```

        description "";
    }
    type inet:ipv4-address;
}
description "";
}
leaf is-rpt {
    type boolean;
    description "";
}

uses multicast-route-attributes;
} // ipv4-route

list ipv6-route {
    when "../.../address-family = 'rt:ipv6'" {
        description
            "Only applicable to ipv4 address family.";
    }
    key "group source-addr is-rpt";
    description "";
    leaf group {
        type inet:ipv6-address;
        description "";
    }
    leaf source-addr {
        type union {
            type enumeration {
                enum '*' {
                    description "";
                }
            }
            type inet:ipv4-address;
        }
        description "";
    }
    leaf is-rpt {
        type boolean;
        description "";
    }

    uses multicast-route-attributes;
} // ipv6-route
} // routes
} // address-family

container interfaces {

```

```
description
  "Containing a list of interfaces.";
list interface {
  key "interface";
  description
    "List of pim interfaces.";
  leaf interface {
    type if:interface-ref;
    description
      "Reference to an entry in the global interface
      list.";
  }
  list address-family {
    key "address-family";
    description
      "Each list entry for one address family.";
    uses rt:address-family;
    uses interface-config-attributes;
    uses interface-state-attributes;
  } // address-family
} // interface
} // interfaces
} // pim
} // augment

/*
 * RPCs
 */

/*
 * Notifications
 */
notification pim-neighbor-event {
  description "Notification event for neighbor.";
  leaf event-type {
    type neighbor-event-type;
    description "Event type.";
  }
  uses pim-neighbor-state-ref;
  leaf up-time {
    type uint32;
    units seconds;
    description "";
  }
}
notification pim-interface-event {
  description "Notification event for interface.";
  leaf event-type {
```

```
    type interface-event-type;
    description "Event type.";
  }
  uses pim-interface-state-ref;
  container ipv4 {
    description "";
    leaf-list address {
      type inet:ipv4-address;
      description "";
    }
    leaf dr-addr {
      type inet:ipv4-address;
      description "";
    }
  }
  container ipv6 {
    description "";
    leaf-list address {
      type inet:ipv6-address;
      description "";
    }
    leaf dr-addr {
      type inet:ipv6-address;
      description "";
    }
  }
}
```

<CODE ENDS>

5.2. PIM RP module

<CODE BEGINS> file "ietf-pim-rp.yang"

```
module ietf-pim-rp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pim-rp";
  // replace with IANA namespace when assigned
  prefix pim-rp;

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

  import ietf-interfaces {
    prefix "if";
  }
}
```

```
}

import ietf-routing {
  prefix "rt";
}

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

organization
  "IETF PIM Working Group";

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

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

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

  Editors:    ";

description
  "The YANG module defines a PIM RP (Rendezvous Point) model.";

revision 2015-12-09 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for PIM";
}

/*
 * Features
 */
feature bsr {
  description
    "This feature indicates that the system supports BSR.";
}

feature bsr-election-state {
  description
    "This feature indicates that the system supports providing
    BSR election state.";
}
```



```
feature static-rp-override {
  description
    "This feature indicates that the system supports configuration
    of static RP override.";
}

feature candidate-interface {
  description
    "This feature indicates that the system supports using
    an interface to configure a BSR or RP candidate.";
}

feature candidate-ipv4 {
  description
    "This feature indicates that the system supports using
    an IPv4 address to configure a BSR or RP candidate.";
}

feature candidate-ipv6 {
  description
    "This feature indicates that the system supports using
    an IPv6 address to configure a BSR or RP candidate.";
}

/*
 * Typedefs
 */
typedef rp-event-type {
  type enumeration {
    enum invalid-jp {
      description
        "An invalid JP message has been received.";
    }
    enum invalid-register {
      description
        "An invalid register message has been received.";
    }
    enum mapping-created {
      description
        "A new mapping has been created.";
    }
    enum mapping-deleted {
      description
        "A mapping has been deleted.";
    }
  }
  description "Operational status event type for notifications.";
}
```

```
}

/*
 * Identities
 */
identity rp-mode {
  description
    "The mode of an RP, which can be SM (Sparse Mode) or
    BIDIR (bi-directional).";
}

identity rp-info-source-type {
  description
    "The information source of an RP.";
}
identity static {
  base rp-info-source-type;
  description
    "The RP is statically configured.";
}
identity bootstrap {
  base rp-info-source-type;
  description
    "The RP is learned from bootstrap.";
}

/*
 * Groupings
 */
grouping bsr-config-attributes {
  description
    "Gouring of BSR config attributes.";
  container bsr-candidate {
    presence
      "Present to serve as a BSR candidate";
    description
      "BSR candidate attributes.";

    choice interface-or-address {
      description
        "Use either interface or ip-address.";
      case interface {
        if-feature candidate-interface;
        leaf interface {
          type if:interface-ref;
          mandatory true;
          description
            "Interface to be used by BSR.";
        }
      }
    }
  }
}
```

```
    }
  }
  case ipv4-address {
    when "../../../../address-family = 'rt:ipv4'" {
      description
        "Only applicable to ipv4 address family.";
    }
    if-feature candidate-ipv4;
    leaf ipv4-address {
      type inet:ipv4-address;
      mandatory true;
      description
        "IP address to be used by BSR.";
    }
  }
  case ipv6-address {
    when "../../../../address-family = 'rt:ipv6'" {
      description
        "Only applicable to ipv6 address family.";
    }
    if-feature candidate-ipv6;
    leaf ipv6-address {
      type inet:ipv6-address;
      mandatory true;
      description
        "IP address to be used by BSR.";
    }
  }
}

leaf hash-mask-length{
  type uint8 {
    range "0..32";
  }
  mandatory true;
  description
    "Value contained in BSR messages used by all routers to
    hash (map) to an RP.";
}

leaf priority {
  type uint8 {
    range "0..255";
  }
  mandatory true;
  description
    "BSR election priority among different candidate BSRs.
    A larger value has a higher priority over a smaller
```

```
        value.";
    }
} // bsr-candidate

list rp-candidate-interface {
    if-feature candidate-interface;
    key "interface";
    description
        "A list of RP candidates";
    leaf interface {
        type if:interface-ref;
        description
            "Interface that the RP candidate uses.";
    }
    uses rp-candidate-attributes;
}

list rp-candidate-ipv4-address {
    when "../../../address-family = 'rt:ipv4'" {
        description
            "Only applicable to ipv4 address family.";
    }
    if-feature candidate-ipv4;
    key "ipv4-address";
    description
        "A list of RP candidates";
    leaf ipv4-address {
        type inet:ipv4-address;
        description
            "IPv4 address that the RP candidate uses.";
    }
    uses rp-candidate-attributes;
}

list rp-candidate-ipv6-address {
    when "../../../address-family = 'rt:ipv6'" {
        description
            "Only applicable to ipv6 address family.";
    }
    if-feature candidate-ipv6;
    key "ipv6-address";
    description
        "A list of RP candidates";
    leaf ipv6-address {
        type inet:ipv6-address;
        description
            "IPv6 address that the RP candidate uses.";
    }
}
```

```
    uses rp-candidate-attributes;
  }
} // bsr-config-attributes

grouping bsr-state-attributes {
  description
    "Gouring of BSR state attributes.";
  container bsr {
    description
      "BSR information.";
    leaf addr {
      type inet:ip-address;
      description "BSR address";
    }
    leaf hash-mask-length {
      type uint8;
      description "";
    }
    leaf priority {
      type uint8 {
        range "0..255";
      }
      description "";
    }
    leaf up-time {
      type uint32;
      units seconds;
      description "";
    }
  }
}
choice election-state {
  if-feature bsr-election-state;
  description "BSR election state.";
  case candidate {
    leaf candidate-bsr-state {
      type enumeration {
        enum "candidate" {
          description
            "The router is a candidate to be the BSR for the
            scope zone, but currently another router is the
            preferred BSR.";
        }
        enum "pending" {
          description
            "The router is a candidate to be the BSR for the
            scope zone. Currently, no other router is the
            preferred BSR, but this router is not yet the
            elected BSR. This is a temporary state that
```

```
        prevents rapid thrashing of the choice of BSR
        during BSR election.";
    }
    enum "elected" {
        description
            "The router is the elected BSR for the scope zone
            and it must perform all the BSR functions.";
    }
}
description
    "Candidate-BSR state.";
reference
    "RFC5059, Section 3.1.1.";
}
}
case "non-candidate" {
    leaf non-candidate-bsr-state {
        type enumeration {
            enum "no-info" {
                description
                    "The router has no information about this scope
                    zone.";
            }
            enum "accept-any" {
                description
                    "The router does not know of an active BSR, and will
                    accept the first Bootstrap message it sees as giving
                    the new BSR's identity and the RP-Set.";
            }
            enum "accept" {
                description
                    "The router knows the identity of the current BSR,
                    and is using the RP-Set provided by that BSR. Only
                    Bootstrap messages from that BSR or from a C-BSR
                    with higher weight than the current BSR will be
                    accepted.";
            }
        }
    }
    description
        "Non-candidate-BSR state.";
    reference
        "RFC5059, Section 3.1.2.";
}
}
} // election-state
leaf bsr-next-bootstrap {
    type uint16;
    units seconds;
}
```

```
    description "";
  }

  container rp {
    description
      "State information of the RP.";
    leaf rp-address {
      type inet:ip-address;
      description "";
    }
    leaf group-policy {
      type string;
      description "";
    }
    leaf up-time {
      type uint32;
      units seconds;
      description "";
    }
  }
  leaf rp-candidate-next-advertisement {
    type uint16;
    units seconds;
    description "";
  }
} // bsr-state-attributes

grouping rp-mapping-state-attributes {
  description
    "Gouring of RP mapping attributes.";
  leaf up-time {
    type uint32;
    units seconds;
    description "";
  }
  leaf expire {
    type pim-base:timer-value;
    description "";
  }
} // rp-mapping-state-attributes

grouping rp-state-attributes {
  description
    "Gouring of RP state attributes.";
  leaf info-source-type {
    type identityref {
      base rp-info-source-type;
    }
  }
}
```

```
        description "";
    } // info-source-type
    leaf up-time {
        type uint32;
        units seconds;
        description "";
    }
    leaf expire {
        type pim-base:timer-value;
        description "";
    }
} // rp-state-attributes

grouping static-rp-attributes {
    description
        "Gouing of static RP attributes, used in augmenting modules.";
    leaf policy-name {
        type string;
        description
            "Static RP policy.";
    }
    leaf override {
        if-feature static-rp-override;
        type boolean;
        description
            "When there is a conflict between static RP and dynamic
            RP, setting this attribute to 'true' will ask the
            system to use static RP.";
    }
} // static-rp-attributes

grouping static-rp-container {
    description
        "Gouing of static RP container.";
    container static-rp {
        description
            "Containing static RP attributes.";
        list ipv4-rp {
            when "../.../address-family = 'rt:ipv4'" {
                description
                    "Only applicable to ipv4 address family.";
            }
            key "ipv4-addr";
            description
                "A list of IPv4 RP addresses.";
            leaf ipv4-addr {
                type inet:ipv4-address;
                description

```



```

        "Specifies a static RP address.";
    }
}

list ipv6-rp {
    when "../.../address-family = 'rt:ipv6'" {
        description
            "Only applicable to ipv6 address family.";
    }
    key "ipv6-addr";
    description
        "A list of IPv6 RP addresses.";
    leaf ipv6-addr {
        type inet:ipv6-address;
        description
            "Specifies a static RP address.";
    }
}
} // static-rp
} // static-rp-container

grouping rp-candidate-attributes {
    description
        "Gouring of RP candidate attributes.";
    leaf policy {
        type string;
        description
            "ACL policy used to filter group addresses.";
    }
    leaf mode {
        type identityref {
            base rp-mode;
        }
        description
            "RP mode.";
    }
}
} // rp-candidate-attributes

/*
 * Configuration data nodes
 */

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family" {
    description "PIM RP augmentation.";

    container rp {

```

```
description
  "PIM RP configuration data.";

uses static-rp-container;

container bsr {
  if-feature bsr;
  description
    "Containing BSR (BootStrap Router) attributes.";
  uses bsr-config-attributes;
} // bsr
} // rp
} // augment

/*
 * Operational state data nodes
 */

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family" {
  description
    "PIM SM state.";

  container rp {
    description
      "PIM RP state data.";

    uses static-rp-container;

    container bsr {
      if-feature bsr;
      description
        "Containing BSR (BootStrap Router) attributes.";
      uses bsr-config-attributes;
      uses bsr-state-attributes;
    } // bsr

    container rp-list {
      description
        "Containing a list RPs.";
      list ipv4-rp {
        when "../.../address-family = 'rt:ipv4'" {
          description
            "Only applicable to ipv4 address family.";
        }
        key "ipv4-addr mode";
        description

```

```
        "A list of IPv4 RP addresses.";
    leaf ipv4-addr {
        type inet:ipv4-address;
        description
            "RP address.";
    }
    leaf mode {
        type identityref {
            base rp-mode;
        }
        description
            "RP mode.";
    }
    leaf info-source-addr {
        type inet:ipv4-address;
        description
            "The address where RP information is learned.";
    }
    uses rp-state-attributes;
}

list ipv6-rp {
    when "../../../address-family = 'rt:ipv6'" {
        description
            "Only applicable to ipv6 address family.";
    }
    key "ipv6-addr mode";
    description
        "A list of IPv6 RP addresses.";
    leaf ipv6-addr {
        type inet:ipv6-address;
        description
            "RP address.";
    }
    leaf mode {
        type identityref {
            base rp-mode;
        }
        description
            "RP mode.";
    }
    leaf info-source-addr {
        type inet:ipv6-address;
        description
            "The address where RP information is learned.";
    }
    uses rp-state-attributes;
}
```

```
    } // rp-list

    container rp-mappings {
      description
        "Containing a list group-to-RP mappings.";
      list ipv4-rp {
        when "../../../address-family = 'rt:ipv4'" {
          description
            "Only applicable to ipv4 address family.";
        }
        key "group rp-addr";
        description
          "A list of group-to-RP mappings.";
        leaf group {
          type inet:ipv4-prefix;
          description
            "Group prefix.";
        }
        leaf rp-addr {
          type inet:ipv4-address;
          description
            "RP address.";
        }
        uses rp-mapping-state-attributes;
      }

      list ipv6-rp {
        when "../../../address-family = 'rt:ipv6'" {
          description
            "Only applicable to ipv6 address family.";
        }
        key "group rp-addr";
        description
          "A list of IPv6 RP addresses.";
        leaf group {
          type inet:ipv6-prefix;
          description
            "Group prefix.";
        }
        leaf rp-addr {
          type inet:ipv6-address;
          description
            "RP address.";
        }
        uses rp-mapping-state-attributes;
      }
    } // rp-mappings
  } // rp
```

```
    } // augment

    /*
     * RPCs
     */

    /*
     * Notifications
     */
    notification pim-rp-event {
        description "Notification event for RP.";
        leaf event-type {
            type rp-event-type;
            description "Event type.";
        }
        uses pim-base:pim-instance-af-state-ref;
        leaf group {
            type inet:ip-address;
            description "";
        }
        leaf rp-address {
            type inet:ip-address;
            description "";
        }
        leaf is-rpt {
            type boolean;
            description "";
        }
        leaf mode {
            type pim-base:pim-mode;
            description "";
        }
        leaf message-origin {
            type inet:ip-address;
            description "";
        }
    }
}
```

<CODE ENDS>

5.3. PIM-SM module

```
<CODE BEGINS> file "ietf-pim-sm.yang"

module ietf-pim-sm {
```

```
namespace "urn:ietf:params:xml:ns:yang:ietf-pim-sm";
// replace with IANA namespace when assigned
prefix pim-sm;

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

import ietf-routing {
  prefix "rt";
}

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

import ietf-pim-rp {
  prefix "pim-rp";
}

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:    ";

description
  "The YANG module defines a sparse mode PIM model.";

revision 2015-12-09 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for PIM";
}

/*
 * Features
 */
```

```
feature spt-switch-infinity {
  description
    "This feature indicates that the system supports configuration
    choice whether to trigger the switchover from the rpt to the
    spt.";
}

feature spt-switch-policy {
  description
    "This feature indicates that the system supports configuring
    policy for the switchover from the rpt to the spt.";
}

/*
 * Identities
 */
identity sm {
  base pim-rp:rp-mode;
  description
    "SM (Spars Mode).";
}

/*
 * Groupings
 */
grouping af-sm-container {
  description
    "Gouping of address family SM container.";
  container sm {
    description
      "PIM SM configuration data.";

    container asm {
      description
        "ASM (Any Source Multicast) attributes.";

      container anycast-rp {
        presence
          "Present to enable anycast RP.";
        description
          "Anycast RP attributes.";

        container ipv4 {
          when "../.../.../address-family = 'rt:ipv4'" {
            description
              "Only applicable to ipv4 address family.";
          }
          description

```

```
    "IPv4 attributes. Only applicable when
    pim-base:address-family is ipv4.";
  list ipv4-anycast-rp {
    key "anycast-addr rp-addr";
    description
      "A list of anycast RP setttings.";
    leaf anycast-addr {
      type inet:ipv4-address;
      description
        "IP address of the anycast RP set. This IP address
        is used by the multicast groups or sources to join
        or register.";
    }

    leaf rp-addr {
      type inet:ipv4-address;
      description
        "IP address of the router configured with anycast
        RP. This is the IP address where the Register
        messages are forwarded.";
    }
  }
}
container ipv6 {
  when "../.../.../address-family = 'rt:ipv6'" {
    description
      "Only applicable to ipv6 address family.";
  }
  description
    "IPv6 attributes. Only applicable when
    pim-base:address-family is ipv6.";
  list ipv6-anycast-rip {
    key "anycast-addr rp-addr";
    description
      "A list of anycast RP setttings.";
    leaf anycast-addr {
      type inet:ipv6-address;
      description
        "IP address of the anycast RP set. This IP address
        is used by the multicast groups or sources to join
        or register.";
    }

    leaf rp-addr {
      type inet:ipv6-address;
      description
        "IP address of the router configured with anycast
        RP. This is the IP address where the Register
```



```

        messages are forwarded.";
    }
}

container spt-switch {
  description
    "SPT (Shortest Path Tree) switching attributes.";
  container infinity {
    if-feature spt-switch-infinity;
    presence "Present if spt-switch is set to infinity.";
    description
      "The receiver's dr never triggers the
       switchover from the rpt to the spt.";
    leaf policy-name {
      if-feature spt-switch-policy;
      type string;
      description
        "Switch policy.";
    }
  } // infinity
}
} // asm

container ssm {
  presence
    "Present to enable SSM (Source-Specific Multicast).";
  description
    "SSM (Source-Specific Multicast) attributes.";

  leaf range-policy {
    type string;
    description
      "Policy used to define SSM address range.";
  }
} // ssm
} // sm
} // af-sm-container

grouping interface-sm-container {
  description
    "Grouping of interface SM container.";
  container sm {
    presence "Present to enable sparse-mode.";
    description
      "PIM SM configuration data.";
  }
}

```

```
    leaf passive {
        type empty;
        description
            "Specifies that no PIM messages are sent out of the PIM
            interface, but the interface can be included in a multicast
            forwarding entry.";
    }
} // sm
} // interface-sm-container

grouping static-rp-sm-container {
    description
        "Grouping that contains SM attributes for static RP.";
    container sm {
        presence
            "Indicate the support of sparse mode.";
        description
            "PIM SM configuration data.";

        uses pim-rp:static-rp-attributes;
    } // sm
} // static-rp-sm-container

/*
 * Configuration data nodes
 */

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family" {
    description "PIM SM augmentation.";

    uses af-sm-container;
} // augment

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:interfaces/pim-base:interface/"
+ "pim-base:address-family" {
    description "PIM SM augmentation.";

    uses interface-sm-container;
} // augment

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp/"
+ "pim-rp:static-rp/pim-rp:ipv4-rp" {
```

```
    description "PIM SM augmentation.";

    uses static-rp-sm-container;
} // augment

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp/"
+ "pim-rp:static-rp/pim-rp:ipv6-rp" {
    description "PIM SM augmentation.";

    uses static-rp-sm-container;
} // augment

/*
 * Operational state data nodes
 */

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family" {
    description
        "PIM SM state.";

    uses af-sm-container;
} // augment

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:interfaces/pim-base:interface/"
+ "pim-base:address-family" {
    description "PIM SM augmentation.";

    uses interface-sm-container;
} // augment

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp/"
+ "pim-rp:static-rp/pim-rp:ipv4-rp" {
    description "PIM SM augmentation.";

    uses static-rp-sm-container;
} // augment

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp/"
```

```
+ "pim-rp:static-rp/pim-rp:ipv6-rp" {
  description "PIM SM augmentation.";

  uses static-rp-sm-container;
} // augment

/*
 * RPCs
 */

/*
 * Notifications
 */
}

<CODE ENDS>
```

5.4. PIM-DM module

```
<CODE BEGINS>

module ietf-pim-dm {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pim-dm";
  // replace with IANA namespace when assigned
  prefix pim-dm;

  import ietf-routing {
    prefix "rt";
  }

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

  organization
    "IETF PIM Working Group";

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

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

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

```
    Editors:    ";

description
  "The YANG module defines a DM (Dense Mode) PIM model.";

revision 2015-10-08 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for PIM";
}

/*
 * Features
 */

/*
 * Identities
 */

/*
 * Groupings
 */

/*
 * Configuration data nodes
 */

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family" {
  description "PIM DM augmentation.";

  container dm {
    presence "Present to enable dense-mode.";
    description
      "PIM DM configuration data.";
  } // Dm
} // augment

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:interfaces/pim-base:interface/"
+ "pim-base:address-family" {
  description "PIM DM augmentation to PIM base interface.";

  container dm {
    presence "Present to enable dense-mode.";
```

```

        description
            "PIM DM configuration data.";
    } // sm
} // augment

/*
 * Operational state data nodes
 */

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family" {
    description
        "PIM DM state.";
    container dm {
        description
            "PIM DM state data.";
    } // dm
} // augment

/*
 * RPCs
 */

/*
 * Notifications
 */
}
<CODE ENDS>

```

5.5. PIM-BIDIR module

```

<CODE BEGINS> file "ietf-pim-bidir.yang"

module ietf-pim-bidir {
    namespace "urn:ietf:params:xml:ns:yang:ietf-pim-bidir";
    // replace with IANA namespace when assigned
    prefix pim-bidir;

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

    import ietf-interfaces {
        prefix "if";
    }

```

```
}

import ietf-routing {
  prefix "rt";
}

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

import ietf-pim-rp {
  prefix "pim-rp";
}

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:    ";

description
  "The YANG module defines a BIDIR (Bidirectional) mode PIM
  model.";

revision 2015-12-22 {
  description
    "Initial revision.";
  reference
    "RFC XXXX: A YANG Data Model for PIM";
}

/*
 * Features
 */
feature intf-df-election {
  description
    "Support configuration of interface DF election.";
}
```

```
/*
 * Identities
 */
identity bidir {
  base pim-rp:rp-mode;
  description
    "BIDIR (Bidirectional) mode.";
}

identity df-state {
  description
    "DF election state type.";
  reference
    "RFC5015: Bidirectional Protocol Independent Multicast
    (BIDIR-PIM).";
}

identity df-state-offer {
  base df-state;
  description
    "Initial election state..";
}

identity df-state-lose {
  base df-state;
  description
    "There either is a different election winner or that no
    router on the link has a path to the RPA.";
}

identity df-state-win {
  base df-state;
  description
    "The router is the acting DF without any contest.";
}

identity df-state-backoff {
  base df-state;
  description
    "The router is the acting DF but another router has made a
    bid to take over..";
}

/*
 * Typedefs
 */

/*
```



```
* Groupings
*/
grouping static-rp-bidir-container {
  description
    "Grouping that contains BIDIR attributes for static RP.";
  container bidir {
    presence
      "Indicate the support of BIDIR mode.";
    description
      "PIM BIDIR configuration data.";

    uses pim-rp:static-rp-attributes;
  } // bidir
} // static-rp-bidir-container

/*
* Configuration data nodes
*/

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family" {
  description "PIM BIDIR augmentation.";

  container bidir {
    description
      "PIM BIDIR configuration data.";
  } // bidir
} // augment

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:interfaces/pim-base:interface/"
+ "pim-base:address-family" {
  description "PIM BIDIR augmentation.";

  container bidir {
    presence "Present to enable BIDIR mode.";
    description
      "PIM BIDIR configuration data.";

    container df-election {
      if-feature intf-df-election;
      description
        "DF election attributes.";
      leaf offer-interval {
        type pim-base:timer-value;
        description "Offer interval";
      }
    }
  }
}
```

```

    }
    leaf backoff-interval {
        type pim-base:timer-value;
        description "Backoff interval";
    }
    leaf offer-multiplier {
        type uint8;
        description "Offer multiplier";
    }
} // df-election
} // bidir
} // augment

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp/"
+ "pim-rp:static-rp/pim-rp:ipv4-rp" {
    description "PIM BIDIR augmentation.";

    uses static-rp-bidir-container;
} // augment

augment "/rt:routing/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp/"
+ "pim-rp:static-rp/pim-rp:ipv6-rp" {
    description "PIM BIDIR augmentation.";

    uses static-rp-bidir-container;
} // augment

/*
 * Operational state data nodes
 */

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family" {
    description
        "PIM BIDIR state.";

    container bidir {
        description
            "PIM BIDIR state data.";
    } // bidir
} // augment

augment "/rt:routing-state/rt:routing-instance/"

```

```
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:interfaces/pim-base:interface/"
+ "pim-base:address-family" {
description "PIM BIDIR augmentation.";

container bidir {
  presence "Present to enable BIDIR mode.";
  description
    "PIM BIDIR configuration data.";

  container df-election {
    if-feature intf-df-election;
    description
      "DF election attributes.";
    leaf offer-interval {
      type pim-base:timer-value;
      description "Offer interval";
    }
    leaf backoff-interval {
      type pim-base:timer-value;
      description "Backoff interval";
    }
    leaf offer-multiplier {
      type uint8;
      description "Offer multiplier";
    }
  } // df-election
} // bidir
} // augment

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp/"
+ "pim-rp:static-rp/pim-rp:ipv4-rp" {
description "PIM BIDIR augmentation.";

  uses static-rp-bidir-container;
} // augment

augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp/"
+ "pim-rp:static-rp/pim-rp:ipv6-rp" {
description "PIM BIDIR augmentation.";

  uses static-rp-bidir-container;
} // augment
```

```
augment "/rt:routing-state/rt:routing-instance/"
+ "rt:routing-protocols/pim-base:pim/"
+ "pim-base:address-family/pim-rp:rp" {
  description "PIM BIDIR augmentation.";

  container bidir {
    description
      "PIM BIDIR state data.";
    container df-election {
      description
        "DF election data.";
      list ipv4-rp {
        when "../../../address-family = 'rt:ipv4'" {
          description
            "Only applicable to ipv4 address family.";
        }
        key "ipv4-addr";
        description
          "A list of IPv4 RP addresses.";
        leaf ipv4-addr {
          type inet:ipv4-address;
          description
            "The address of the RP.";
        }
      } // ipv4-rp
      list ipv6-rp {
        when "../../../address-family = 'rt:ipv6'" {
          description
            "Only applicable to ipv6 address family.";
        }
        key "ipv6-addr";
        description
          "A list of IPv6 RP addresses.";
        leaf ipv6-addr {
          type inet:ipv6-address;
          description
            "The address of the RP.";
        }
      } // ipv6-rp
    } // df-election
  }

  container interface-df-election {
    description
      "Interface DF election data.";
    list ipv4-rp {
      when "../../../address-family = 'rt:ipv4'" {
        description
          "Only applicable to ipv4 address family.";
      }
    }
  }
}
```

```
    }
    key "ipv4-addr interface-name";
    description
      "A list of IPv4 RP addresses.";
    leaf ipv4-addr {
      type inet:ipv4-address;
      description
        "The address of the RP.";
    }
    leaf interface-name {
      type if:interface-ref;
      description
        "The address of the RP.";
    }
    leaf df-address {
      type inet:ipv4-address;
      description
        "DF address.";
    }
    leaf interface-state {
      type identityref {
        base df-state;
      }
      description
        "Interface state.";
    }
  } // ipv4-rp
  list ipv6-rp {
    when "../.../.../address-family = 'rt:ipv6'" {
      description
        "Only applicable to ipv6 address family.";
    }
    key "ipv6-addr interface-name";
    description
      "A list of IPv6 RP addresses.";
    leaf ipv6-addr {
      type inet:ipv6-address;
      description
        "The address of the RP.";
    }
    leaf interface-name {
      type if:interface-ref;
      description
        "The address of the RP.";
    }
    leaf df-address {
      type inet:ipv6-address;
      description
```

```
        "DF address.";
    }
    leaf interface-state {
        type identityref {
            base df-state;
        }
        description
            "Interface state.";
    }
    } // ipv6-rp
    } // interface-df-election
    }
} // augment

/*
 * RPCs
 */

/*
 * Notifications
 */
}
```

<CODE ENDS>

6. TODO list

In this draft, several aspects of the model are incomplete. The PIM-DM and BI-DIR modules are just placeholders for now to demonstrate the hierarchy of the model. Other things the draft will include when complete but does not yet include:

- o Interaction with BFD protocol
- o Constraints (constant ranges, validation)
- o State-limiting and policy
- o Statistics.

For these subjects, we will employ the same design principles of expressing a highly general model; vendors may use features and add augmentations in order to express which subsets of this general model are valid in their implementations.

7. Security Considerations

The data model defined does not introduce any security implications.

This draft does not change any underlying security issues inherent in [I-D.ietf-netmod-routing-cfg].

8. IANA Considerations

TBD

9. Acknowledgements

The authors would like to thank Steve Baillargeon, Guo Feng, Hu Fangwei, Robert Kebler, Tanmoy Kundu, Liu Yisong, Mahesh Sivakumar, and Stig Venaas for their valuable contributions.

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, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<http://www.rfc-editor.org/info/rfc6020>>.
- [RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <<http://www.rfc-editor.org/info/rfc6241>>.
- [I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-05 (work in progress), October 2015.

10.2. Informative References

- [RFC3569] Bhattacharyya, S., Ed., "An Overview of Source-Specific Multicast (SSM)", RFC 3569, DOI 10.17487/RFC3569, July 2003, <<http://www.rfc-editor.org/info/rfc3569>>.

- [RFC3973] Adams, A., Nicholas, J., and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", RFC 3973, DOI 10.17487/RFC3973, January 2005, <<http://www.rfc-editor.org/info/rfc3973>>.
- [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>>.
- [RFC4610] Farinacci, D. and Y. Cai, "Anycast-RP Using Protocol Independent Multicast (PIM)", RFC 4610, DOI 10.17487/RFC4610, August 2006, <<http://www.rfc-editor.org/info/rfc4610>>.
- [RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <<http://www.rfc-editor.org/info/rfc5015>>.
- [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, <<http://www.rfc-editor.org/info/rfc5059>>.
- [RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <<http://www.rfc-editor.org/info/rfc6087>>.
- [I-D.ietf-netmod-routing-cfg]
Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-20 (work in progress), October 2015.

Authors' Addresses

Liu Xufeng
Ericsson
1595 Spring Hill Road, Suite 500
Vienna VA 22182
USA

EMail: xufeng.liu@ericsson.com

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

EMail: pete.mcallister@metaswitch.com

Anish Peter
Juniper Networks
Electra, Exora Business Park
Bangalore, KA 560103
India

EMail: anishp@juniper.net

IP Multicast
Internet-Draft
Intended status: Informational
Expires: June 26, 2016

E. Vyncke
Cisco
E. Rey
ERNW
A. Atlasis
NCI Agency
December 24, 2015

MLD Security
draft-vyncke-pim-mld-security-01

Abstract

The latest version of Multicast Listener Discovery protocol is defined in RFC 3810, dated back in 2004, while the first version of MLD, which is still in use and has not been deprecated, is defined in RFC 2710 and is dated back in 1999. New security research has exhibited new vulnerabilities in MLD, both remote and local attack vectors. This document describes those vulnerabilities and proposes specific mitigation techniques.

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 June 26, 2016.

Copyright Notice

Copyright (c) 2015 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. Local Vulnerabilities	3
2.1. Downgrading to MLDv1	3
2.2. Queries sent to unicast address	4
2.3. Win the election	4
2.4. Host enumeration and OS fingerprinting	4
2.5. Flooding of MLD messages	4
2.6. Amplification	4
3. Remote Vulnerabilities	5
4. Mitigations	5
5. IANA Considerations	5
6. Security Considerations	6
7. Acknowledgements	6
8. References	6
8.1. Normative References	6
8.2. Informative References	6
Authors' Addresses	7

1. Introduction

The Multicast Listener Discovery protocol version 2 (MLDv2) RFC3810 [RFC3810] has a security section but it was not exhaustive and the focus was only on local forged MLD packets. The same is also true for the first version of MLD (now called MLDv1), which is still in use, defined in RFC 2710. This document goes beyond those attacks.

For the reader who is not familiar with MLDv2, here are the main points:

Multicast routers send MLD queries which are either generic (query about all multicast group) sent to ff02::1 (link-scope all nodes) or specific (query about a specific group) sent to this multicast group. Query messages can also be sent to a unicast address.

Multicast members reply to MLDv2 queries with reports sent to ff02::16 (link-scope all MLDv2 routers). In version 1 of MLD RFC2710 [RFC2710], the reports are sent to the multicast group being reported. Reports can be transmitted twice or more in order to ensure that the MLD router gets at least one report.

When a node ceases to listen to a multicast address on an interface, it sends an MLDv1 Done message or a specially crafted MLDv2 Report message.

All MLD packets are ICMPv6 RFC4443 [RFC4443] messages sent with a hop-limit of 1, from a link-local address and there is no authentication.

MLD messages received with a hop-limit greater than 1 should be discarded.

Neighbor Discovery Protocol RFC4861 [RFC4861] requires nodes to become member of the respective solicited-node multicast groups for all their link-scope and global-scope addresses.

Switches are assumed to implement MLD snooping RFC4541 [RFC4541] to learn where to forward multicast packets. It must be noted though that implementations of MLD snooping do not act on link-local multicast groups such as solicited-node multicast group: they simply forward all packets destined to a link-local multicast group to all port in the same layer-2 network.

MLDv2 was designed to be interoperable with MLDv1.

The main difference between MLDv1 and MLDv2 from a functionality perspective is that MLDv1 does not support "source filtering" (in MLDv2 nodes can report interest in traffic only from a set of source addresses or from all except a set source addresses).

Every IPv6 node must support MLD.

This document is heavily based on previous research: [Troopers2015].

2. Local Vulnerabilities

2.1. Downgrading to MLDv1

A single MLDv1 report message is enough to downgrade all MLD nodes (hosts and routers) to the version 1 protocol. This could be used to force a MLD host to reply with MLDv1 reports sent to the multicast group rather than to ff02::16. This downgrade to MLDv1 could also be used to transmit the MLDv1 report with a 'done' operation to remove a listener (stopping the multicast traffic on the subnet). Another consequence of downgrading to MLDv1 can be the fact that an attacker can also used "Host Suppression" feature as part of a DoS attack, make the launch of such an attack easier.

2.2. Queries sent to unicast address

Section 5.1.15 of RFC3810 [RFC3810], specifies that for debugging purposes, nodes must accept and process queries sent to any of their addresses (including unicast). Lab testing, described in [Troopers2015], clearly shows that all implementations except FreeBSD accept and process MLD queries sent to a unicast global address. This can be exploited to completely bypass the legitimate MLD router and interact directly (for whatever purpose) with the targets (including legitimate routers and clients).

2.3. Win the election

When there are multiple MLD routers in a layer-2 domain, the one with the lowest IPv6 address wins the election and becomes the designated MLD router. A hostile node can then send from a lower link-local address an MLD message and become the MLD router. This fact in combination with the direct interaction with the targets could be leveraged to mount a denial of service attack.

2.4. Host enumeration and OS fingerprinting

Some hosts try to prevent host enumeration by not responding to ICMPv6 echo request messages sent to any multicast group. But, the same hosts must reply to any MLD queries including the generic one sent to ff02::1, this allows for MLD host enumeration. As hosts join different groups based on their operating system (specific groups for Microsoft Windows for example), the MLD report can also help for Operating System (OS) fingerprinting.

2.5. Flooding of MLD messages

If an implementation does not rate limit in hardware the rate of processed MLD messages, then they are vulnerable to a CPU exhaustion denial of services. If a node does not limit the number of states associated to MLD, then this node is vulnerable to a memory exhaustion denial of services.

2.6. Amplification

Nodes usually join multiple groups (for example, Microsoft Windows 8.1 joins 4 groups). Therefore a forged generic MLDv1 query will force those nodes to transmit MLDv1 reports for each of their groups (in our example 4); furthermore, many implementations send MLD reports twice (in our example 8 in total). MLDv2 is a little better because reports are sent to ff02::16 and not to the multicast group.

3. Remote Vulnerabilities

MLD messages with hop-limit different than 1 should be discarded but nothing prevents a hostile party located n hops away from the victim to send any MLD messages with a hop-limit set to $n+1$. Therefore, a remote hostile party can mount attacks against MLD (especially because implementations process MLD queries sent to a global unicast address).

4. Mitigations

This section proposes some mitigation techniques that could be used to prevent the above attacks. This section is not a specification of any kind, the words 'should' is plain English and is not related to RFC2119 [RFC2119].

Mitigation by specific implementations:

Similar to RA-guard RFC6105 [RFC6105], there should be a MLD-guard function in layer-2 switches; MLD queries (either version 1 or version 2) received on ports attached to non multicast routers should be discarded. Switches could also block all MLDv1 packets in order to prevent the downgrading of MLD version. Of course, this requires all nodes to support MLDv2.

All nodes should be able to disable MLDv1.

Control plane policing should also be implemented in order to avoid denial of services attacks.

Mitigation by a protocol update of RFC2710 [RFC2710] and RFC3810 [RFC3810]:

MLD queries should not be accepted and processed when sent to a unicast address (either link-local or global scope). This requires update of RFC 3810 and RFC 2710.

To mitigate the remote attacks, the hop-limit should have been set to 255 and MLD nodes should discard packets with a hop-limit different than 255.

5. IANA Considerations

This document contains no IANA considerations.

6. Security Considerations

This document describes multiple vulnerabilities that have been described above and tries to mitigate them or even eliminate some of them by making specific suggestions for update of the protocol as well as by suggesting the implementation of related security mechanisms to layer-2 devices.

7. Acknowledgements

The authors would like to thank Stig Venaas for some discussions on this topic.

8. References

8.1. Normative References

- [RFC2710] Deering, S., Fenner, W., and B. Haberman, "Multicast Listener Discovery (MLD) for IPv6", RFC 2710, DOI 10.17487/RFC2710, October 1999, <<http://www.rfc-editor.org/info/rfc2710>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<http://www.rfc-editor.org/info/rfc3810>>.

8.2. Informative 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>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification", RFC 4443, DOI 10.17487/RFC4443, March 2006, <<http://www.rfc-editor.org/info/rfc4443>>.
- [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, <<http://www.rfc-editor.org/info/rfc4541>>.

- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<http://www.rfc-editor.org/info/rfc4861>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<http://www.rfc-editor.org/info/rfc6105>>.
- [Troopers2015] Rey, E., Atlasis, A., and J. Salazar, "MLD Considered Harmful", 2015, <https://www.troopers.de/media/filer_public/7c/35/7c35967a-d0d4-46fb-8a3b-4c16df37ce59/troopers15_ipv6secsummit_atlasis_rey_salazar_mld_considered_harmful_final.pdf>.

Authors' Addresses

Eric Vyncke
Cisco
De Kleetlaan 6a
Diegem 1831
Belgium

Phone: +32 2 778 4677
Email: evyncke@cisco.com

Enno Rey
ERNW
Carl-Bosch-Str. 4
Heidelberg 69115
Germany

Phone: +49 6221 480390
Email: erey@ernw.de

Antonios Atlasis
NCI Agency
Oude Waalsdorperweg 61
The Hague 2597 AK
The Netherlands

Phone: +31 703743564
Email: antonios.atlasis@ncia.nato.int