SPRING Working Group
Internet Draft
Intended status: Standards Track
Expires: March 2017

Dave Allan
Ericsson
Jeff Tantsura

September 2016

A Framework for Computed Multicast applied to MPLS based Segment Routing
draft-allan-spring-mpls-multicast-framework-02

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.

Table of Contents

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. Changes from the last version

Clarifications in the pruning and simplification algorithm

3. Conventions used in this document

3.1. Terminology

Candidate replication point (CRP) - 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 is employed to determine and install the required state to implement the segment. For MPLS a multicast segment is implemented as a p2mp LSP. A multicast segment is identified by a multicast SID.

Multicast SID - Is the data plane identifier that is used to implement a multicast segment. As per a unicast MPLS segment, the rightmost 20 bits of a multicast SID is encoded as a label. It is drawn from an SRGB that is global to the SR domain.

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.

4. 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 the operator 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 the operator 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).

4.1. 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 multicast segment is assigned a SID, typically by management configuration of the node that will be the overall root for the source specific tree. The root node then uses the IGP to advertise this information to all nodes in the IGP area/domain.

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.

4.2. Role of the Routing System

The role of the IGP is to communicate topology information, multicast capability and associated algorithm, multicast registrations, unicast to SID bindings, multicast to SID bindings and waypoints in multi-segment MDTs. No changes to topology or unicast to SID binding advertisements 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.

4.3. 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).

4.4. 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.

5. Elements of Procedure

5.1. 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.

5.2. FIB Determination

5.2.1. 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 identified by the waypoint SID 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.

An example of this would be consider a node "x", and another node "y". At some point in time, "x" advertises a tree that identifies "y" as a waypoint that cross connects upstream SID "a" to downstream SID "b". At some later point node "y" fails. The other nodes in the network will compute segment "a" as if it included all leaves and waypoints in segment "b". All apriori state installed for segment "b"

would be removed as the failure of "y" has required "b" to be subsumed by "a".

5.2.2. 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:

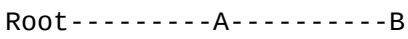
All possible shortest paths to the set of leaves for the MDT is determined. Then pruning rules are repeatedly applied until no further prunes are possible.

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:

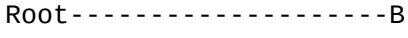
1) Eliminate any links and nodes not on a potential shortest path from the root to the leaves for the MDT under consideration.

2) Simplify via the replacement of any nodes that do not have a potential role in the MDT with links.

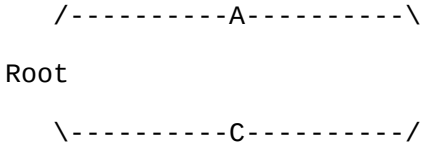
This will be nodes that are not a leaf, a root or a candidate replication point. For example:



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.



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

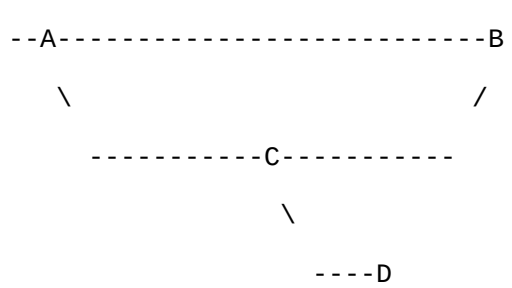


Where A and C have no role and the cost root-A-B = cost root-C-B, they can be replaced with a single link from Root to B.

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:



Link AB is cost 2, link AC and CB are cost 1 (cost of link CD does not affect the example).

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 upstream links.

The normal procedure is to determine the closest upstream leaf or pinned path and then compare all upstream adjacencies with that metric

a. If the upstream adjacency extends closer to the root than the closest leaf or pinned path, then that adjacency can be pruned.

b. If the upstream adjacency extends the same distance towards the root then

i. If it is to a non-leaf or pinned path candidate replication point, it can be pruned

ii. If it is to a pinned path, where there are equal upstream adjacencies that terminate on leaves, it can be pruned (considered inferior).

iii. If there is more than one "equal" upstream adjacency, that is all terminate on nodes that are on pinned paths, or all terminate on nodes that are leaves, than one is selected. This is via the lowest node ID.

c. If the upstream adjacency is a candidate replication point closer than the closest leaf, and upstream from it is a node that is a leaf or pinned path equidistant with the closest leaf, then all adjacencies that extend to leaves ranked lower than the leaf or pinned path behind the CRP may be pruned. Note that an upstream adjacency that has a CRP closer than the closest leaf or pinned path cannot be pruned.

d. When for a given node all possible upstream adjacencies that can be pruned have been identified, each is removed, and any simplifications that can be performed as a result of the prune are performed. This is the equivalent of a localized check for 2 and 3 above and is then performed iteratively in response to changes to the graph as a result of pruning.

The procedure is to implement 1, 2 and 3 above, then loop on 4 until such time as the MDT is fully resolved, or no further prunes are possible. Step 4 is performed in a specific order. The nodes are processed according to a ranking from closest to the root to the farthest, and from lowest node ID to the highest within a given distance from the root.

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.

5.3. 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.

5.4. 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 are network wide configurable values.

6. Related work

6.1. IGP Extensions

The required IGP changes are documented in [MCAST-ISIS] and [MCAST-OPSF].

6.2. 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

7. 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.

8. Acknowledgements

Thanks to Uma Chunduri for his detailed review and suggestions.

9. Security Considerations

For a future version of this document.

10. IANA Considerations

This document requires the allocation of a PMSI tunnel type to identify a SPRING P2MP tunnel type from the P-Multicast Service Interface Tunnel (PMSI Tunnel) Tunnel Types registry.

11. References

11.1. Normative References

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

11.2. Informative References

[MCAST-ISIS] Allan et.al., "IS-IS extensions for Computed Multicast applied to MPLS based Segment Routing", IETF work in progress, draft-allan-isis-spring-multicast-00, July 2016

[MCAST-OSPF] Allan et.al., "OSPF extensions for Computed Multicast applied to MPLS based Segment Routing", IETF work in progress, draft-allan-ospf-spring-multicast-00, July 2016

[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

12. Authors' Addresses

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

Jeff Tantsura
Email: jefftant.ietf@gmail.com

PIM Working Group
Internet-Draft
Intended Status: Standard Track
Expires: April 28, 2017

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

October 28, 2016

A YANG data model for Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD)
draft-ietf-pim-igmp-mld-yang-01.txt

Status of this Memo

Copyright Notice

Abstract

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

Table of Contents

1. Introduction

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 augments "/rt:routing/rt:control-plane-protocols" as opposed to augmenting "/rt:routing/rt:control-plane-protocols/ rt:control-plane-protocol" as the latter would allow multiple protocol instances per VRF, which does not make sense for IGMP and MLD.

```
augment /rt:routing/rt:control-plane-protocols:
   +--rw igmp
      |  +--rw global
      |  |  +--rw enable?        boolean {global-admin-enable}?
      |  |  +--rw max-entries?   uint32 {global-max-entries}?
      |  |  +--rw max-groups?    uint32 {global-max-groups}?
      |  +--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-ale
rt}?
      |     +--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-source
s}?
      |        +--rw query-interval?                uint16
      |        +--rw query-max-response-time?       uint16
      |        +--rw require-router-alert?          boolean {intf-require-router-
alert}?
      |        +--rw robustness-variable?           uint8
      |        +--rw source-policy?                 string {intf-source-policy}?
      |        +--rw verify-source-subnet?          empty {intf-verify-source-sub
net}?
      |        +--rw version?                       uint8
      |        +--rw join-group*                    inet:ipv4-address {intf-join-
group}?
      |        +--rw ssm-map* [source-addr group-policy] {intf-ssm-map}?
      |        |  +--rw source-addr     ssm-map-ipv4-addr-type
      |        |  +--rw group-policy    string
```

```
      |        +--rw static-group* [group 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-ale
rt}?
            +--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-source
s}?
               +--rw query-interval?                uint16
               +--rw query-max-response-time?       uint16
               +--rw require-router-alert?          boolean {intf-require-router-
alert}?
               +--rw robustness-variable?           uint8
               +--rw source-policy?                 string {intf-source-policy}?
               +--rw verify-source-subnet?          empty {intf-verify-source-sub
net}?
               +--rw version?                       uint8
               +--rw join-group*                    inet:ipv6-address {intf-join-
group}?
               +--rw ssm-map* [source-addr group-policy] {intf-ssm-map}?
               |  +--rw source-addr     ssm-map-ipv6-addr-type
               |  +--rw group-policy    string
               +--rw static-group* [group source-addr] {intf-static-group}?
                  +--rw group          inet:ipv6-address
                  +--rw source-addr    source-ipv6-addr-type
```

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:control-plane-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-ale
rt}?
      |     +--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-source
      |        +--ro query-interval?              uint16
      |        +--ro query-max-response-time?     uint16
      |        +--ro require-router-alert?        boolean {intf-require-router-
alert}?
      |        +--ro robustness-variable?         uint8
      |        +--ro source-policy?               string {intf-source-policy}?
      |        +--ro verify-source-subnet?        empty {intf-verify-source-sub
net}?
      |        +--ro version?                     uint8
      |        +--ro join-group*                  inet:ipv4-address {intf-join-
group}?
      |        +--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-
group}?
      |        +--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-ale
rt}?
            +--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-source
s}?
               +--ro query-interval?               uint16
               +--ro query-max-response-time?      uint16
               +--ro require-router-alert?         boolean {intf-require-router-
alert}?
               +--ro robustness-variable?          uint8
               +--ro source-policy?                string {intf-source-policy}?
               +--ro verify-source-subnet?         empty {intf-verify-source-sub
net}?
               +--ro version?                      uint8
               +--ro join-group*                   inet:ipv6-address {intf-join-
group}?
               +--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-
group}?
               +--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 interface?   -> /rt:routing/control-plane-protocols/igmp/inter
faces/interface/interface

      |     +---w group?       inet:ipv4-address

      |     +---w source?      inet:ipv4-address

      +---x clear-mld-groups {rpc-clear-groups}?

         +---w input

            +---w interface?   -> /rt:routing/control-plane-protocols/mld/interf
aces/interface/interface

            +---w group?       inet:ipv6-address

            +---w source?      inet:ipv6-address
```

4. IGMP and MLD YANG Modules

```
   <CODE BEGINS> file "ietf-igmp-mld@2016-10-28.yang"
   module ietf-igmp-mld {
      namespace "urn:ietf:params:xml:ns:yang:ietf-igmp-mld";
      // replace with IANA namespace when assigned
      prefix igmp-mld;

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

      import ietf-yang-types {
```

```
    prefix "yang";
  }

  import ietf-routing {
    prefix "rt";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-ip {
    prefix ip;
  }

  organization
    "IETF PIM Working Group";

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

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

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

     Editors:   ";

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

  revision 2016-10-28 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for IGMP and MLD";
  }

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  feature 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-mld;

        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 igmp 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-mld

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

      uses interface-state-attributes-igmp-mld;

      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-mld;
          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:control-plane-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:control-plane-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 interface {
            type leafref {
              path "/rt:routing/rt:control-plane-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 IPv4 address.
               If it is not specified, all IGMP group tables are
               cleared.";
          }
          leaf source {
            type inet:ipv4-address;
            description
              "Multicast source IPv4 address.
               If it is not specified, all IGMP source-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 interface {
            type leafref {
              path "/rt:routing/rt:control-plane-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:ipv6-address;
            description
```

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

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

5. 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].

6. IANA Considerations

TBD

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.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, October 2010

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, June 2011

[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.

8. Acknowledgments

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

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: May 4, 2017

IJ. Wijnands
S. Venaas
Cisco Systems, Inc.
M. Brig
Aegis BMD Program Office
A. Jonasson
Swedish Defence Material Administration (FMV)
October 31, 2016

PIM flooding mechanism and source discovery
draft-ietf-pim-source-discovery-bsr-05

Abstract

PIM Sparse-Mode uses a Rendezvous Point 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 protcol that provides a way to support PIM Sparse Mode (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.

Table of Contents

1. Introduction

PIM Sparse-Mode 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 Sparse Mode (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 mechanism is defined in this document, and is modeled after the Bootstrap Router mechanism [RFC5059]. 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.

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-SA: PFM Source Announcement.

SG Mapping: Multicast source to group mapping.

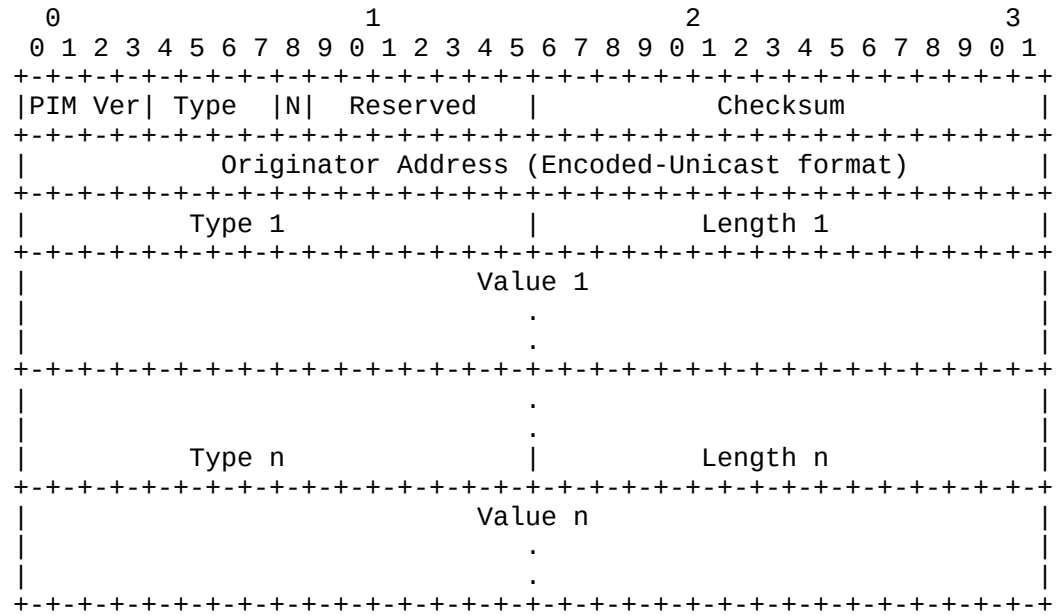
2. Testing and deployment experiences

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. In this network with frequent topology changes and link or router failures, PIM-SM with RP election is 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 message. For the tests, all routers were configured to send PFM-SA 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. We are defining a more generic mechanism that can flood any kind of information throughout a PIM domain. It is not necessarily a domain though, it depends on the administrative boundaries being configured. The forwarding rules are identical to BSR, except that there is no BSR election and 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. Just like BSR it is also sent hop by hop. Note that there is no built in election mechanism as in BSR, so there can be multiple originators. We call this mechanism the PIM Flooding Mechanism (PFM).

3.1. PFM message format



PIM Version: Reserved, Checksum Described 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.

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].

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.

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

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

3.2. 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.

3.2.1. Initial checks

In order to do further processing, a message MUST meet the following requirements. The message MUST be from a directly connected neighbor for which we have active Hello state, and it MUST have been sent to the ALL-PIM-ROUTERS group. Also, the interface MUST NOT be an administrative boundary for PFM. If No-Forward is not set, it MUST have been sent by the RPF neighbor for the originator address. If No-Forward is set, we MUST have restarted within 60 seconds. In pseudo-code the algorithm is as follows:

```
if ((DirectlyConnected(PFM.src_ip_address) == FALSE) OR
    (we have no Hello state for PFM.src_ip_address) 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 startup)) {
    drop the message silently, optionally log error.
}
```

3.2.2. Processing and forwarding of PFM messages

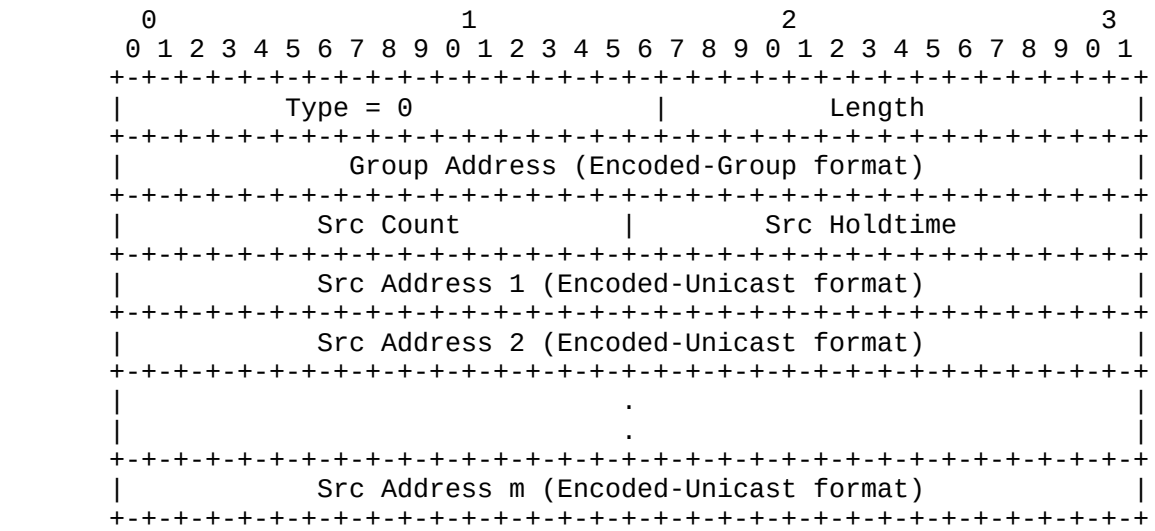
When the message is received, the initial checks above must be performed. If it passes the checks, we then for each included TLV perform processing according to the specification for that TLV.

After processing, we forward the message. Unless otherwise specified by the type specification, the TLVs in the forwarded message are identical to the TLVs in the received message. However, if the most significant bit in the type field is set (the type value is larger than 32767) and we do not support the type, then that particular type should be omitted from the forwarded messages. The message is forwarded out of all interfaces with PIM neighbors (including the interface it was received on).

4. Distributing Source to Group Mappings

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

4.1. Group Source Holdtime TLV



Type: This TLV has type 0.

Length: The length of the value.

Group Address: The group we are announcing sources for. The format for this address is given in the Encoded-Group format in [RFC7761].

Src Count: How many unicast encoded sources address encodings follow.

Src Holdtime: The Holdtime (in seconds) for the corresponding 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 PFM messages

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 router originating the PFM messages includes one of its own addresses in the originator field. Note that this address SHOULD be routeable due to RPF checking. The PFM messages containing the GSH TLV are periodically sent for as long as the multicast source is active, similar to how PIM registers are periodically sent. The default announcement period is 60 seconds, which means that as long as the source is active, it is included in a PFM message originated every 60 seconds. The holdtime for the source is by default 210 seconds. Other values MAY be configured, but the holdtime MUST be either zero, or larger than the announcement period. It is RECOMMENDED to be 3.5 times the announcement period. 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.

4.3. Processing GSH TLVs

A router that receives a PFM message containing GSH TLVs SHOULD parse the message and store each of the GSH TLVs as SG mappings with a holdtimer started with the advertised holdtime. 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. 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) we recommend using PIM Bidir [RFC5015] or SSM [RFC4607]. 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).

Another solution to address the problems described above is documented in [I-D.ietf-magma-msnip]. This proposal allows for a host to tell the FHR its willingness to act as Source for a certain Group before sending the data packets. LHRs have time to join the SPT before the host starts sending which would avoid packet loss. The SG mappings announced by [I-D.ietf-magma-msnip] can be advertised directly in SG messages, allowing a nice integration of both proposals. The life time of the SPT is not driven by the liveliness of Multicast data packets (which is the case with PIM SM), but by the announcements driven via [I-D.ietf-magma-msnip]. This will also prevent packet loss due to bursty sources.

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. As soon as the network heals, the SG Mappings are re-flooded into the other partition(s) and other receivers can join to the newly learned sources.

5. Security Considerations

The security considerations are mainly similar to what is documented in [RFC5059]. It is a concern that rogue devices can inject packets that are flooded throughout a domain. PFM packets must only be accepted from a PIM neighbor. Deployments may use mechanisms for authenticating PIM neighbors. For PFM-SA it is an issue that injected packets from a rogue device could send SG mappings for a large number of source addresses, causing routers to use memory storing these mappings, and also if they have interest in the groups, build Shortest Path Trees for sources that are not actually active.

6. IANA considerations

This document requires the assignment of a new PIM message type for the PIM Flooding Mechanism (PFM). IANA is also requested to create a registry for PFM TLVs, with type 0 assigned to the "Source Group Holdtime" TLV. Values in the range 1-65535 are "Unassigned". Assignments for the registry are to be made according to the policy "IETF Review" as defined in [RFC5226].

7. Acknowledgments

The authors would like to thank Arjen Boers for contributing to the initial idea, and Yiqun Cai and Dino Farinacci for their comments on the draft.

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>.

[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>.

[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>.

8.2. Informative References

[RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <http://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, <http://www.rfc-editor.org/info/rfc5015>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[I-D.ietf-magma-msnip] Fenner, B., Haberman, B., Holbrook, H., Kouvelas, I., and S. Venaas, "Multicast Source Notification of Interest Protocol (MSNIP)", draft-ietf-magma-msnip-06 (work in progress), March 2011.

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: April 14, 2017

X. Liu
Kuatro Technologies
P. McAllister
Metaswitch Networks
A. Peter
Juniper Networks
M. Sivakumar
Cisco Systems
Y. Liu
Huawei Technologies
F. Hu
ZTE Corporation

October 11, 2016

A YANG data model for Protocol-Independent Multicast (PIM)
draft-ietf-pim-yang-03

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 April 14, 2017.

Copyright Notice

Table of Contents

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 [RFC7761], 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 data model can be used to configure and manage these protocol features. The operational state data and statistics can be retrieved by this model. The protocol specific notifications are also defined in the model.

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 specified in separate documents.

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

This document 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 other protocol Yang models such as IS-IS.

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. Module Structure

3.1. 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. This data model agrees to a routing-instance-centric (VRF) model view 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 augments "/rt:routing/rt:control-plane-protocols" as opposed to augmenting "/rt:routing/rt:control-plane-protocols/ rt:control-plane-protocol" as the latter would allow multiple protocol instances per VRF, which does not make sense for PIM.

```
          module: ietf-pim-base
augment /rt:routing/rt:control-plane-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?           bfd-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:control-plane-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 df-election?                  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 df-election?                  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 df-election?                  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-address is-rpt]
|     |  +--ro group                 inet:ipv4-address
|     |  +--ro source-address        union
|     |  +--ro is-rpt                boolean
|     |  +--ro expiration?           timer-value
|     |  +--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 expiration?   timer-value
|     |     +--ro up-time?      uint32
|     |     +--ro jp-state?     enumeration
|     +--ro ipv6-route* [group source-address is-rpt]
|        +--ro group                 inet:ipv6-address
|        +--ro source-address        union
|        +--ro is-rpt                boolean
```

```
         |        +--ro expiration?           timer-value
         |        +--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 expiration?   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?           bfd-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-address?   inet:ipv4-address
                  +--ro ipv6
                  |  +--ro address*      inet:ipv6-address
                  |  +--ro dr-address?   inet:ipv6-address
                  +--ro oper-status?         enumeration
                  +--ro hello-expiration?    timer-value
                  +--ro neighbor-ipv4* [address]
                  |  +--ro address        inet:ipv4-address
                  |  +--ro bfd-status?    enumeration
                  |  +--ro expiration?    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 expiration?    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 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 interface-state-ref?   leafref
         +--ro ipv4
         |  +--ro address*      inet:ipv4-address
         |  +--ro dr-address?   inet:ipv4-address
         +--ro ipv6
            +--ro address*      inet:ipv6-address
            +--ro dr-address?   inet:ipv6-address
```

3.2. 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:control-plane-protocols/pim-base:pim/
```

```
   pim-base:address-family:
      +--rw rp
         +--rw static-rp
         |  +--rw ipv4-rp* [ipv4-address]
         |  |  +--rw ipv4-address    inet:ipv4-address
         |  +--rw ipv6-rp* [ipv6-address]
         |     +--rw ipv6-address    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:control-plane-protocols/pim-base:pim/
   pim-base:address-family:
      +--ro rp
         +--ro static-rp
         |  +--ro ipv4-rp* [ipv4-address]
         |  |  +--ro ipv4-address    inet:ipv4-address
         |  +--ro ipv6-rp* [ipv6-address]
         |     +--ro ipv6-address    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 address?            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-address mode]
        |  |  +--ro ipv4-address           inet:ipv4-address
        |  |  +--ro mode                   identityref
        |  |  +--ro info-source-address?   inet:ipv4-address
        |  |  +--ro info-source-type?      identityref
        |  |  +--ro up-time?               uint32
        |  |  +--ro expiration?            pim-base:timer-value
        |  +--ro ipv6-rp* [ipv6-address mode]
        |     +--ro ipv6-address           inet:ipv6-address
        |     +--ro mode                   identityref
        |     +--ro info-source-address?   inet:ipv6-address
        |     +--ro info-source-type?      identityref
```

```
         |     +--ro up-time?              uint32
         |     +--ro expiration?           pim-base:timer-value
         +--ro rp-mappings
            +--ro ipv4-rp* [group rp-address]
            |  +--ro group         inet:ipv4-prefix
            |  +--ro rp-address    inet:ipv4-address
            |  +--ro up-time?      uint32
            |  +--ro expiration?   pim-base:timer-value
            +--ro ipv6-rp* [group rp-address]
               +--ro group         inet:ipv6-prefix
               +--ro rp-address    inet:ipv6-address
               +--ro up-time?      uint32
               +--ro expiration?   pim-base:timer-value
   notifications:
      +---n pim-rp-event
         +--ro event-type?              rp-event-type
         +--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
```

3.3. PIM-SM module

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

```
           module: ietf-pim-sm
   augment /rt:routing/rt:control-plane-protocols/pim-base:pim/
   pim-base:address-family:
      +--rw sm
         +--rw asm
         |  +--rw anycast-rp!
         |  |  +--rw ipv4
         |  |  |  +--rw ipv4-anycast-rp* [anycast-address rp-address]
         |  |  |     +--rw anycast-address    inet:ipv4-address
         |  |  |     +--rw rp-address         inet:ipv4-address
         |  |  +--rw ipv6
         |  |     +--rw ipv6-anycast-rp* [anycast-address rp-address]
         |  |        +--rw anycast-address    inet:ipv6-address
         |  |        +--rw rp-address         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:control-plane-protocols/pim-base:pim/
pim-base:interfaces/pim-base:interface/pim-base:address-family:
   +--rw sm!
      +--rw passive?   empty
augment /rt:routing/rt:control-plane-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:control-plane-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:control-plane-protocols/pim-base:pim/
pim-base:address-family:
   +--ro sm
      +--ro asm
      |  +--ro anycast-rp!
      |  |  +--ro ipv4
      |  |  |  +--ro ipv4-anycast-rp* [anycast-address rp-address]
      |  |  |     +--ro anycast-address    inet:ipv4-address
      |  |  |     +--ro rp-address         inet:ipv4-address
      |  |  +--ro ipv6
      |  |     +--ro ipv6-anycast-rp* [anycast-address rp-address]
      |  |        +--ro anycast-address    inet:ipv6-address
      |  |        +--ro rp-address         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:control-plane-protocols/pim-base:pim/
pim-base:interfaces/pim-base:interface/pim-base:address-family:
   +--ro sm!
      +--ro passive?   empty
augment /rt:routing-state/rt:control-plane-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:control-plane-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}?
```

3.4. PIM-DM module

This module will cover Dense Mode configuration.

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

3.5. PIM-BIDIR module

This module will cover Bidirectional PIM configuration.

```
          module: ietf-pim-bidir
augment /rt:routing/rt:control-plane-protocols/pim-base:pim/
pim-base:address-family:
   +--rw bidir!
augment /rt:routing/rt:control-plane-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:control-plane-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:control-plane-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:control-plane-protocols/pim-base:pim/
pim-base:address-family:
   +--ro bidir
augment /rt:routing-state/rt:control-plane-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:control-plane-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:control-plane-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:control-plane-protocols/pim-base:pim/
   pim-base:address-family/pim-rp:rp:
      +--ro bidir
         +--ro df-election
         |  +--ro ipv4-rp* [ipv4-address]
         |  |  +--ro ipv4-address    inet:ipv4-address
         |  +--ro ipv6-rp* [ipv6-address]
         |     +--ro ipv6-address    inet:ipv6-address
         +--ro interface-df-election
            +--ro ipv4-rp* [ipv4-address interface-name]
            |  +--ro ipv4-address       inet:ipv4-address
            |  +--ro interface-name     if:interface-ref
            |  +--ro df-address?        inet:ipv4-address
            |  +--ro interface-state?   identityref
            +--ro ipv6-rp* [ipv6-address interface-name]
               +--ro ipv6-address       inet:ipv6-address
               +--ro interface-name     if:interface-ref
               +--ro df-address?        inet:ipv6-address
               +--ro interface-state?   identityref
```

4. PIM YANG Modules

4.1. PIM base module

```
   <CODE BEGINS> file "ietf-pim-base@2016-09-22.yang"
   module ietf-pim-base {
     namespace "urn:ietf:params:xml:ns:yang:ietf-pim-base";
     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>

     Editor:   Xufeng Liu
               <mailto:xliu@kuatrotech.com>

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

     Editor:   Anish Peter
               <mailto:anishp@juniper.net>

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

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

     Editor:   Fangwei Hu
               <mailto:hu.fangwei@zte.com.cn>";

  description
```

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

    revision 2016-09-22 {
      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 configuration 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 address family configuration 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 is the number by which the hello
                 interval is multplied to obtain the hold time";
            }
          }
        }
        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 multiplier {
          if-feature intf-jp-multiplier;
          leaf jp-multiplier {
            type uint8;
            description
              "Join prune multiplier is the number by which the join
               prune interval is multplied to obtain the hold time";
          }
        }
      }
      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 "Interface state attributes for IPv4.";
        leaf-list address {
          type inet:ipv4-address;
```

```
          description "List of addresses.";
        }
        leaf dr-address {
          type inet:ipv4-address;
          description "Designated Router (DR) address.";
        }
      }
      container ipv6 {
        when "../address-family = 'rt:ipv6'" {
          description
            "Only applicable to IPv6 address family.";
        }
        description "Interface state attributes for IPv6.";
        leaf-list address {
          type inet:ipv6-address;
          description "List of addresses.";
        }
        leaf dr-address {
          type inet:ipv6-address;
          description "Designated Router (DR) address.";
        }
      }
      uses interface-state-af-attributes;
    } // interface-state-attributes

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

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

      leaf hello-expiration {
        type timer-value;
        description "Hello interval expiration time.";
      }
```

```
      list neighbor-ipv4 {
        when "../address-family = 'rt:ipv4'" {
          description
            "Only applicable to IPv4 address family.";
        }
        key "address";
        description "Neighbor state information.";
        leaf address {
          type inet:ipv4-address;
          description "Neighbor address.";
        }
        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 "Neighbor state information.";
        leaf address {
          type inet:ipv6-address;
          description "Neighbor address.";
        }
        uses neighbor-state-af-attributes;
      } // list neighbor-ipv6
    } // interface-state-af-attributes

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

      leaf expiration {
        type timer-value;
        description "When the route will expire.";
      }
      leaf incoming-interface {
        type if:interface-ref;
        description
          "Reference to an entry in the global interface
           list.";
      }
      leaf mode {
        type pim-mode;
        description "PIM mode.";
      }
      leaf msdp-learned {
```

```
          type boolean;
          description "'true' if route is learned from MSDP.";
        }
        leaf rp-address {
          type inet:ip-address;
          description "RP address.";
        }
        leaf rpf-neighbor {
          type inet:ip-address;
          description "RPF neighbor address.";
        }
        leaf spt-bit {
          type boolean;
          description "'true' if SPT bit is set.";
        }
        leaf up-time {
          type uint32;
          units seconds;
          description "Up time duration.";
        }
        list outgoing-interface {
          key "name";
          description
            "A list of outgoing interfaces.";

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

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

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

          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 "Join-prune state.";
        }
      }
    } // multicast-route-attributes

    grouping neighbor-state-af-attributes {
      description
        "A grouping defining neighbor per address family attributes.";
      leaf bfd-status {
        type enumeration {
          enum up {
            description
              "BFD is up.";
          }
          enum down {
            description
              "BFD is down.";
          }
        }
        description "BFD status.";
      }
      leaf expiration {
        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 "Up time duration.";
      }
    } // 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-af-state-ref {
      description
        "An absolute reference to a PIM instance address family.";
      leaf instance-af-state-ref {
        type leafref {
          path "/rt:routing-state/rt:control-plane-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.";
      leaf interface-state-ref {
        type leafref {
          path "/rt:routing-state/rt:control-plane-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:control-plane-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 "Only applicable to IPv4 address family.";
        }
        type leafref {
          path "/rt:routing-state/rt:control-plane-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 "Only applicable to IPv6 address family.";
        }
        type leafref {
          path "/rt:routing-state/rt:control-plane-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 "A container defining statistics attributes.";
        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 "Containing error statistics.";
          uses statistics-error;
        }
        container queue {
          description "Containing queue statistics.";
          uses statistics-queue;
        }
        container received {
          description "Containing statistics of received messages.";
          uses statistics-sent-received;
        }
        container sent {
          description "Containing statistics of sent messages.";
          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 df-election {
        type yang:counter64;
        description
          "The number of DF election 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:control-plane-protocols" {
      description
        "PIM augmentation to routing instance configuration.";

      container pim {
        presence "Container for PIM protocol.";
        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:control-plane-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 "Containing topology tree information.";
          list ipv4-route {
            when "../../../address-family = 'rt:ipv4'" {
              description
                "Only applicable to IPv4 address family.";
            }
            key "group source-address is-rpt";
            description "A list of IPv4 routes.";
            leaf group {
              type inet:ipv4-address;
              description "Group address.";
            }
            leaf source-address {
              type union {
                type enumeration {
                  enum '*' {
                    description "Use '*' to indicate any address.";
                  }
                }
```

```
                  type inet:ipv4-address;
                }
                description "Source address.";
              }
              leaf is-rpt {
                type boolean;
                description "'true' if the tree is RPT.";
              }

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

            list ipv6-route {
              when "../../../address-family = 'rt:ipv6'" {
                description
                  "Only applicable to IPv6 address family.";
              }
              key "group source-address is-rpt";
              description "A list of IPv6 routes.";
              leaf group {
                type inet:ipv6-address;
                description "Group address.";
              }
              leaf source-address {
                type union {
                  type enumeration {
                    enum '*' {
                      description "Use '*' to indicate any address.";
                    }
                  }
                  type inet:ipv4-address;
                }
                description "Source address.";
              }
              leaf is-rpt {
                type boolean;
                description "'true' if the tree is RPT.";
              }

              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 "Up time duration.";
      }
    }
    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 "Containing IPv4 information.";
         leaf-list address {
           type inet:ipv4-address;
           description "List of addresses.";
         }
         leaf dr-address {
           type inet:ipv4-address;
           description "Designated Router (DR) address.";
         }
       }
       container ipv6 {
         description "Containing IPv6 information.";
         leaf-list address {
           type inet:ipv6-address;
           description "List of addresses.";
         }
         leaf dr-address {
           type inet:ipv6-address;
           description "Designated Router (DR) address.";
         }
       }
     }
   }
   <CODE ENDS>
```

4.2. PIM RP module

```
  <CODE BEGINS> file "ietf-pim-rp@2016-09-22.yang"
  module ietf-pim-rp {
    namespace "urn:ietf:params:xml:ns:yang:ietf-pim-rp";
    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>

       Editor:   Xufeng Liu
                 <mailto:xliu@kuatrotech.com>

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

       Editor:   Anish Peter
                 <mailto:anishp@juniper.net>

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

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

       Editor:   Fangwei Hu
                 <mailto:hu.fangwei@zte.com.cn>";

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

    revision 2016-09-22 {
      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
        "Grouping 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 "../../../pim-base: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 "../../../pim-base: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..128";
          }
          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 "../../../pim-base: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 "../../../pim-base: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
        "Grouping of BSR state attributes.";
      container bsr {
        description
          "BSR information.";
        leaf address {
          type inet:ip-address;
          description "BSR address";
        }
        leaf hash-mask-length {
          type uint8 {
            range "0..128";
          }
          description "Hash mask length.";
        }
        leaf priority {
          type uint8 {
            range "0..255";
          }
          description "Priority.";
        }
        leaf up-time {
          type uint32;
          units seconds;
          description "Up time duration.";
        }
      }
      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 "The time when next bootstrap will be sent.";
    }

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

  grouping rp-mapping-state-attributes {
    description
      "Grouping of RP mapping attributes.";
    leaf up-time {
      type uint32;
      units seconds;
      description "Up time duration.";
    }
```

```
      leaf expiration {
        type pim-base:timer-value;
        description "Expiration time.";
      }
    } // rp-mapping-state-attributes

    grouping rp-state-attributes {
      description
        "Grouping of RP state attributes.";
      leaf info-source-type {
        type identityref {
          base rp-info-source-type;
        }
        description "The information source of an RP.";
      } // info-source-type
      leaf up-time {
        type uint32;
        units seconds;
        description "Up time duration.";
      }
      leaf expiration {
        type pim-base:timer-value;
        description "Expiration time.";
      }
    } // rp-state-attributes

    grouping static-rp-attributes {
      description
        "Grouping 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
        "Grouping of static RP container.";
      container static-rp {
```

```
        description
          "Containing static RP attributes.";
        list ipv4-rp {
          when "../../../pim-base:address-family = 'rt:ipv4'" {
            description
              "Only applicable to IPv4 address family.";
          }
          key "ipv4-address";
          description
            "A list of IPv4 RP addresses.";
          leaf ipv4-address {
            type inet:ipv4-address;
            description
              "Specifies a static RP address.";
          }
        }

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

    grouping rp-candidate-attributes {
      description
        "Grouping 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:control-plane-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:control-plane-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 of RPs.";
          list ipv4-rp {
            when "../../../pim-base:address-family = 'rt:ipv4'" {
              description
                "Only applicable to IPv4 address family.";
            }
            key "ipv4-address mode";
            description
              "A list of IPv4 RP addresses.";
            leaf ipv4-address {
              type inet:ipv4-address;
              description
                "RP address.";
            }
            leaf mode {
              type identityref {
                base rp-mode;
              }
              description
                "RP mode.";
            }
            leaf info-source-address {
              type inet:ipv4-address;
              description
                "The address where RP information is learned.";
            }
            uses rp-state-attributes;
          }

          list ipv6-rp {
            when "../../../pim-base:address-family = 'rt:ipv6'" {
              description
                "Only applicable to IPv6 address family.";
            }
            key "ipv6-address mode";
            description
              "A list of IPv6 RP addresses.";
            leaf ipv6-address {
              type inet:ipv6-address;
              description
                "RP address.";
            }
            leaf mode {
              type identityref {
                base rp-mode;
              }
```

```
            description
              "RP mode.";
          }
          leaf info-source-address {
            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 of group-to-RP mappings.";
        list ipv4-rp {
          when "../../../pim-base:address-family = 'rt:ipv4'" {
            description
              "Only applicable to IPv4 address family.";
          }
          key "group rp-address";
          description
            "A list of group-to-RP mappings.";
          leaf group {
            type inet:ipv4-prefix;
            description
              "Group prefix.";
          }
          leaf rp-address {
            type inet:ipv4-address;
            description
              "RP address.";
          }
          uses rp-mapping-state-attributes;
        }

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

```
            }
            leaf rp-address {
              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 "Group address.";
      }
      leaf rp-address {
        type inet:ip-address;
        description "RP address.";
      }
      leaf is-rpt {
        type boolean;
        description "'true' if the tree is RPT.";
      }
      leaf mode {
        type pim-base:pim-mode;
        description "PIM mode.";
      }
      leaf message-origin {
        type inet:ip-address;
        description "Where the message is originated.";
      }
    }
  }
```

<CODE ENDS>

4.3. PIM-SM module

```
<CODE BEGINS> file "ietf-pim-sm@2016-09-22.yang"
module ietf-pim-sm {
  namespace "urn:ietf:params:xml:ns:yang:ietf-pim-sm";
  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>

     Editor:   Xufeng Liu
               <mailto:xliu@kuatrotech.com>

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

     Editor:   Anish Peter
```

```
                <mailto:anishp@juniper.net>

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

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

        Editor:   Fangwei Hu
                  <mailto:hu.fangwei@zte.com.cn>";

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

    revision 2016-09-22 {
      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 (Sparse Mode).";
    }

    /*
     * Groupings
```

```
      */
     grouping af-sm-container {
       description
         "Grouping 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 "../../../../pim-base: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-address rp-address";
                 description
                   "A list of anycast RP settings.";
                 leaf anycast-address {
                   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-address {
                   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 "../../../../pim-base: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-rp {
              key "anycast-address rp-address";
              description
                "A list of anycast RP settings.";
              leaf anycast-address {
                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-address {
                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 switchover threshold is set to
               infinity, according to the policy specified below.";
            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 or accepted on
             this 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:control-plane-protocols/pim-base:pim/"
       + "pim-base:address-family" {
       description "PIM SM augmentation.";

       uses af-sm-container;
     } // augment

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

       uses af-sm-container;
     } // augment

     augment "/rt:routing-state/rt:control-plane-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:control-plane-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:control-plane-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>
```

4.4. PIM-DM module

```
  <CODE BEGINS> file "ietf-pim-dm@2016-09-22.yang"
  module ietf-pim-dm {
    namespace "urn:ietf:params:xml:ns:yang:ietf-pim-dm";
    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>

       Editor:   Xufeng Liu
                 <mailto:xliu@kuatrotech.com>

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

       Editor:   Anish Peter
                 <mailto:anishp@juniper.net>

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

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

       Editor:   Fangwei Hu
                 <mailto:hu.fangwei@zte.com.cn>";

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

    revision 2016-09-22 {
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for PIM";
    }

    /*
     * Features
     */
```

```
     /*
      * Identities
      */

     /*
      * Groupings
      */

     /*
      * Configuration data nodes
      */

     augment "/rt:routing/rt:control-plane-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:control-plane-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:control-plane-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>
```

4.5. PIM-BIDIR module

```
  <CODE BEGINS> file "ietf-pim-bidir@2016-09-22.yang"
  module ietf-pim-bidir {
    namespace "urn:ietf:params:xml:ns:yang:ietf-pim-bidir";
    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>

           Editor: Xufeng Liu
                   <mailto:xliu@kuatrotech.com>

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

           Editor: Anish Peter
                   <mailto:anishp@juniper.net>

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

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

           Editor: Fangwei Hu
                   <mailto:hu.fangwei@zte.com.cn>";

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

     revision 2016-09-22 {
       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 df-election-container {
      description
        "Grouping that contains DF election attributes.";
      container df-election {
```

```
        if-feature intf-df-election;
        description
          "DF election attributes.";
        leaf offer-interval {
          type pim-base:timer-value;
          description
            "Offer interval specifies the interval between repeated
             DF election messages.";
        }
        leaf backoff-interval {
          type pim-base:timer-value;
          description
            "This is the interval that the acting DF waits between
             receiving a better DF Offer and sending the Pass message
             to transfer DF responsibility";
        }
        leaf offer-multiplier {
          type uint8;
          description
            "This is number of transmission attempts for DF election
             messages.
             When a DF election Offer or Winner message fails to be
             received, the message is retransmitted.
             The offer-multiplier sets the minimum number of DF
             election messages that must fail to be received for DF
             election to fail.
             If a router receives from a neighbor a better offer than
             its own, the router stops participating in the election
             for a period of offer-multiplier * offer-interval.
             Eventually, all routers except the best candidate stop
             sending Offer messages.";
        }
      } // df-election
    } // df-election-container

    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:control-plane-protocols/pim-base:pim/"
       + "pim-base:address-family" {
       description "PIM BIDIR augmentation.";

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

     augment "/rt:routing/rt:control-plane-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.";

         uses df-election-container;
       } // bidir
     } // augment

     augment "/rt:routing/rt:control-plane-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:control-plane-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:control-plane-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:control-plane-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.";

      uses df-election-container;
    } // bidir
  } // augment

  augment "/rt:routing-state/rt:control-plane-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:control-plane-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:control-plane-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 "../../../../pim-base:address-family = 'rt:ipv4'" {
              description
                "Only applicable to IPv4 address family.";
            }
            key "ipv4-address";
            description
              "A list of IPv4 RP addresses.";
            leaf ipv4-address {
              type inet:ipv4-address;
              description
                "The address of the RP.";
            }
          } // ipv4-rp
          list ipv6-rp {
            when "../../../../pim-base:address-family = 'rt:ipv6'" {
              description
                "Only applicable to IPv6 address family.";
            }
            key "ipv6-address";
            description
              "A list of IPv6 RP addresses.";
            leaf ipv6-address {
              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 "../../../../pim-base:address-family = 'rt:ipv4'" {
              description
                "Only applicable to IPv4 address family.";
            }
            key "ipv4-address interface-name";
            description
              "A list of IPv4 RP addresses.";
            leaf ipv4-address {
              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 "../../../../pim-base:address-family = 'rt:ipv6'" {
              description
                "Only applicable to IPv6 address family.";
            }
            key "ipv6-address interface-name";
            description
              "A list of IPv6 RP addresses.";
            leaf ipv6-address {
              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>
```

5. Security Considerations

Configuration and state data defined in this document are designed to be accessed via a management protocol with secure transport layer, such as NETCONF [RFC6241]. The NETCONF access control model [RFC6536] provides the means to restrict access for specific users to a pre- configured subset of all available operations and contents.

The models defined in this document contain a number of configuration data nodes that are writable, creatable, and deletable. Unauthorised access to the configuration data can adversely affect the routing subsystem of both the local device and the network. This may lead to network malfunctions, delivery of packets to inappropriate destinations and other problems.

6. IANA Considerations

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

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

```
 ------------------------------------------------------------------
 URI: urn:ietf:params:xml:ns:yang:ietf-pim-base
 Registrant Contact: The IESG.
 XML: N/A, the requested URI is an XML namespace.
 ------------------------------------------------------------------
```

```
---------------------------------------------------------------------
URI: urn:ietf:params:xml:ns:yang:ietf-pim-bidir
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------

---------------------------------------------------------------------
URI: urn:ietf:params:xml:ns:yang:ietf-pim-dm
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------

---------------------------------------------------------------------
URI: urn:ietf:params:xml:ns:yang:ietf-pim-rp
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------

---------------------------------------------------------------------
URI: urn:ietf:params:xml:ns:yang:ietf-pim-sm
Registrant Contact: The IESG.
XML: N/A, the requested URI is an XML namespace.
---------------------------------------------------------------------
```

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

```
---------------------------------------------------------------------
name:         ietf-pim-base
namespace:    urn:ietf:params:xml:ns:yang:ietf-pim-base
prefix:       pim-base
reference:    RFC XXXX
---------------------------------------------------------------------

---------------------------------------------------------------------
name:         ietf-pim-bidir
namespace:    urn:ietf:params:xml:ns:yang:ietf-pim-bidir
prefix:       pim-bidir
reference:    RFC XXXX
---------------------------------------------------------------------

---------------------------------------------------------------------
name:         ietf-pim-dm
namespace:    urn:ietf:params:xml:ns:yang:ietf-pim-dm
prefix:       pim-dm
reference:    RFC XXXX
---------------------------------------------------------------------
```

```
------------------------------------------------------------------
name:         ietf-pim-rp
namespace:    urn:ietf:params:xml:ns:yang:ietf-pim-rp
prefix:       pim-rp
reference:    RFC XXXX
------------------------------------------------------------------

------------------------------------------------------------------
name:         ietf-pim-sm
namespace:    urn:ietf:params:xml:ns:yang:ietf-pim-sm
prefix:       pim-sm
reference:    RFC XXXX
------------------------------------------------------------------
```

7. Acknowledgements

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

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>.

[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-08 (work in progress), September 2016.

8.2. 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>.

[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>.

[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>.

[I-D.ietf-netmod-routing-cfg] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-23 (work in progress), August 2016.

Authors' Addresses

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

EMail: xliu@kuatrotech.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

Mahesh Sivakumar
Cisco Systems
510 McCarthy Boulevard
Milpitas, California
USA

EMail: masivaku@cisco.com

Yisong Liu
Huawei Technologies
Huawei Administration Building
Longgang, Guangdong 518129
China

EMail: liuyisong@huawei.com

Fangwei Hu
ZTE Corporation
889 Bibo Road
Shanghai, Shanghai 201203
China

EMail: hu.fangwei@zte.com.cn

PIM WG
Internet-Draft
Intended status: Standards Track
Expires: May 1, 2017

X. Liu
Kuatro Technologies
Z. Zhang
ZTE Corporation
A. Peter
Juniper Networks
M. Sivakumar
Cisco Systems
F. Guo
Huawei Technologies
P. McAllister
Metaswitch Networks

October 28, 2016

MSDP YANG Model
draft-zhang-pim-msdp-yang-02

Abstract

This document defines a YANG data model for the configuration and management of MSDP Protocol.

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 May 1, 2017.

Copyright Notice

Table of Contents

1. Introduction

[RFC3618] introduces the protocol definition of MSDP. This document defines a YANG data model that can be used to configure and manage the MSDP protocol. The operational state data and statistics can also be retrieved by this model.

This model is designed to be used along with other multicast YANG models such as PIM, which are not covered in this document.

2. Design of the Data Model

This model imports and augments ietf-routing YANG model defined in [I-D.ietf-netmod-routing-cfg]. Both configuration branch and state branch of [I-D.ietf-netmod-routing-cfg] are augmented. The configuration branch covers global configuration attributes and per peer configuration attributes. The state branch includes global, per peer, and source-active information. The container "msdp" is the top level container in this data model. The presence of this container is expected to enable MSDP protocol functionality.

```
module: ietf-msdp
augment /rt:routing/rt:control-plane-protocols:
   +--rw msdp!
      +--rw global
      |  +--rw connect-source?   if:interface-ref
      |  +--rw default-peer! {global-default-peer}?
      |  |  +--rw peer-addr        leafref
```

```
      |  |  +--rw prefix-policy?   string {global-default-peer-policy}?
      |  +--rw originating-rp
      |  |  +--rw interface?   if:interface-ref
      |  +--rw sa-filter
      |  |  +--rw in?    string
      |  |  +--rw out?   string
      |  +--rw sa-limit?         uint32 {global-sa-limit}?
      |  +--rw ttl-threshold?    uint8
      +--rw peers
         +--rw peer* [address]
            +--rw address           inet:ipv4-address
            +--rw authentication
            |  +--rw (authentication-type)?
            |     +--:(key-chain) {peer-key-chain}?
            |     |  +--rw key-chain?                key-chain:key-chain-ref
            |     +--:(password) {peer-key-chain}?
            |        +--rw key?                      string
            |        +--rw (algorithm)?
            |           +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
            |           |  +--rw hmac-sha1-12?             empty
            |           +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
            |           |  +--rw aes-cmac-prf-128?         empty
            |           +--:(md5)
            |           |  +--rw md5?                      empty
            |           +--:(sha-1)
            |           |  +--rw sha-1?                    empty
            |           +--:(hmac-sha-1)
            |           |  +--rw hmac-sha-1?               empty
            |           +--:(hmac-sha-256)
            |           |  +--rw hmac-sha-256?             empty
            |           +--:(hmac-sha-384)
            |           |  +--rw hmac-sha-384?             empty
            |           +--:(hmac-sha-512)
            |           |  +--rw hmac-sha-512?             empty
            |           +--:(clear-text) {clear-text}?
            |           |  +--rw clear-text?               empty
            |           +--:(replay-protection-only) {replay-protection-only}?
            |              +--rw replay-protection-only?   empty
            +--rw enable?           boolean {peer-admin-enable}?
            +--rw connect-source?   if:interface-ref
            +--rw description?      string {peer-description}?
            +--rw mesh-group?       string
            +--rw peer-as?          string {peer-as}?
            +--rw sa-filter
            |  +--rw in?    string
            |  +--rw out?   string
            +--rw sa-limit?         uint32 {peer-sa-limit}?
            +--rw timer
```

```
            |  +--rw connect-retry-interval?   uint16 {peer-timer-connect-retry}
?
            |  +--rw holdtime-interval?        uint16 {peer-timer-holdtime}?
            |  +--rw keepalive-interval?       uint16 {peer-timer-keepalive}?
            +--rw ttl-threshold?    uint8
augment /rt:routing-state/rt:control-plane-protocols:
   +--ro msdp!
      +--ro global
      |  +--ro connect-source?   if:interface-ref
      |  +--ro default-peer! {global-default-peer}?
      |  |  +--ro peer-addr        leafref
      |  |  +--ro prefix-policy?   string {global-default-peer-policy}?
      |  +--ro originating-rp
      |  |  +--ro interface?   if:interface-ref
      |  +--ro sa-filter
      |  |  +--ro in?    string
      |  |  +--ro out?   string
      |  +--ro sa-limit?         uint32 {global-sa-limit}?
      |  +--ro ttl-threshold?    uint8
      +--ro peers
      |  +--ro peer* [address]
      |     +--ro address                inet:ipv4-address
      |     +--ro authentication
      |     |  +--ro (authentication-type)?
      |     |     +--:(key-chain) {peer-key-chain}?
      |     |     |  +--ro key-chain?               key-chain:key-chain-ref
      |     |     +--:(password) {peer-key-chain}?
      |     |        +--ro key?                     string
      |     |        +--ro (algorithm)?
      |     |           +--:(hmac-sha-1-12) {crypto-hmac-sha-1-12}?
      |     |           |  +--ro hmac-sha1-12?             empty
      |     |           +--:(aes-cmac-prf-128) {aes-cmac-prf-128}?
      |     |           |  +--ro aes-cmac-prf-128?         empty
      |     |           +--:(md5)
      |     |           |  +--ro md5?                      empty
      |     |           +--:(sha-1)
      |     |           |  +--ro sha-1?                    empty
      |     |           +--:(hmac-sha-1)
      |     |           |  +--ro hmac-sha-1?               empty
      |     |           +--:(hmac-sha-256)
      |     |           |  +--ro hmac-sha-256?             empty
      |     |           +--:(hmac-sha-384)
      |     |           |  +--ro hmac-sha-384?             empty
      |     |           +--:(hmac-sha-512)
      |     |           |  +--ro hmac-sha-512?             empty
      |     |           +--:(clear-text) {clear-text}?
      |     |           |  +--ro clear-text?               empty
      |     |           +--:(replay-protection-only) {replay-protection-only}?
      |     |              +--ro replay-protection-only?   empty
```

```
      |     +--ro enable?                boolean {peer-admin-enable}?
      |     +--ro connect-source?        if:interface-ref
      |     +--ro description?           string {peer-description}?
      |     +--ro mesh-group?            string
      |     +--ro peer-as?               string {peer-as}?
      |     +--ro sa-filter
      |     |  +--ro in?    string
      |     |  +--ro out?   string
      |     +--ro sa-limit?              uint32 {peer-sa-limit}?
      |     +--ro timer
      |     |  +--ro connect-retry-interval?   uint16 {peer-timer-connect-retry}
?
      |     |  +--ro holdtime-interval?        uint16 {peer-timer-holdtime}?
      |     |  +--ro keepalive-interval?       uint16 {peer-timer-keepalive}?
      |     +--ro ttl-threshold?         uint8
      |     +--ro session-state?         enumeration
      |     +--ro elapsed-time?          uint32
      |     +--ro connect-retry-expire?  uint32
      |     +--ro hold-expire?           uint32
      |     +--ro is-default-peer?       boolean
      |     +--ro keepalive-expire?      uint32
      |     +--ro reset-count?           uint32
      |     +--ro statistics
      |        +--ro discontinuity-time?   yang:date-and-time
      |        +--ro error
      |        |  +--ro rpf-failure?   uint32
      |        +--ro queue
      |        |  +--ro size-in?    uint32
      |        |  +--ro size-out?   uint32
      |        +--ro received
      |        |  +--ro keepalive?      yang:counter64
      |        |  +--ro notification?   yang:counter64
      |        |  +--ro sa-message?     yang:counter64
      |        |  +--ro sa-response?    yang:counter64
      |        |  +--ro sa-request?     yang:counter64
      |        |  +--ro total?          yang:counter64
      |        +--ro sent
      |           +--ro keepalive?      yang:counter64
      |           +--ro notification?   yang:counter64
      |           +--ro sa-message?     yang:counter64
      |           +--ro sa-response?    yang:counter64
      |           +--ro sa-request?     yang:counter64
      |           +--ro total?          yang:counter64
      +--ro sa-cache
         +--ro entry* [group source-addr]
            +--ro group               inet:ipv4-address
            +--ro source-addr         union
            +--ro origin-rp* [rp-address]
            |  +--ro rp-address       inet:ip-address
```

```
            |  +--ro is-local-rp?     boolean
            |  +--ro sa-adv-expire?   uint32
            +--ro up-time?             uint32
            +--ro expire?              uint32
            +--ro holddown-interval?   uint32
            +--ro peer-learned-from?   inet:ipv4-address
            +--ro rpf-peer?            inet:ipv4-address
rpcs:
   +---x msdp-clear-peer
   |  +---w input
   |     +---w peer-address?   inet:ipv4-address
   +---x msdp-clear-sa-cache {rpc-clear-sa-cache}?
      +---w input
         +---w entry!
         |  +---w group          inet:ipv4-address
         |  +---w source-addr?   union
         +---w peer-address?   inet:ipv4-address
         +---w peer-as?        string
```

3. MSDP configuration

MSDP configurations require peer configurations. Several peers may be configured in a mesh-group. The Source-Active information may be filtered by peers.

The configuration modeling branch is composed of MSDP global and peer configurations. The two parts are the most important parts of MSDP.

Besides the fundamental features of MSDP protocol, several optional features are included in the model. These features help the control of MSDP protocol. The peer features and SA features make the deployment and control easier. The connection parameters can be used to control the TCP connection because MSDP protocol is based on TCP. The authentication features make the protocol more secure. The filter features allow operators to avoid the irrelevant information.

4. MSDP State

MSDP states are composed of MSDP global state, MSDP peer state, statistics information and Sa-cache information. The statistics information and Sa-cache information helps the operator to retrieve the protocol condition.

5. MSDP RPC

The part is used to define some useful and ordinary operations of protocol management.

6. Notifications

This part will be updated in later version.

7. MSDP YANG model

```
<CODE BEGINS> file "ietf-msdp@2016-10-18.yang"
module ietf-msdp {
  namespace "urn:ietf:params:xml:ns:yang:ietf-msdp";
  prefix msdp;

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

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

  import ietf-routing {
    prefix "rt";
  }

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-ip {
    prefix "ip";
  }

  import ietf-key-chain {
    prefix "key-chain";
  }

  organization
    "IETF PIM( Protocols for IP Multicast ) 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 the YANG definitions for MSDP.";

  revision 2016-10-18 {
    description
      "Initial revision.";
    reference
      "RFC XXXX: A YANG Data Model for MSDP.
       RFC 3618: Multicast Source Discovery Protocol (MSDP).
       RFC 4624: Multicast Source Discovery Protocol (MSDP) MIB";
  }

  /*
   * Features
   */
  feature global-connect-source {
    description
      "Support configuration of global connect-source.";
  }

  feature global-default-peer {
    description
      "Support configuration of global default peer.";
  }

  feature global-default-peer-policy {
    description
      "Support configuration of global default peer.";
  }

  feature global-sa-filter {
    description
      "Support configuration of global SA filter.";
  }

  feature global-sa-limit {
    description
      "Support configuration of global limit on SA entries.";
  }

  feature global-ttl-threshold {
    description
      "Support configuration of global ttl-threshold.";
  }

  feature rpc-clear-sa-cache {
    description
      "Support the rpc to clear SA cache.";
```

```
    }

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

    feature peer-as {
      description
        "Support configuration of peer AS number.";
    }

    feature peer-connect-source {
      description
        "Support configuration of global connect-source.";
    }

    feature peer-description {
      description
        "Support configuration of peer description.";
    }

    feature peer-key-chain {
      description
        "Support configuration of peer key-chain.";
    }

    feature peer-password {
      description
        "Support configuration of peer key-chain.";
    }

    feature peer-sa-limit {
      description
        "Support configuration of per peer limit on SA entries.";
    }

    feature peer-timer-connect-retry {
      description
        "Support configuration of peer timer for connect-retry.";
    }

    feature peer-timer-keepalive {
      description
        "Support configuration of peer timer for keepalive.";
    }

    feature peer-timer-holdtime {
```

```
      description
        "Support configuration of peer timer for holdtime.";
    }

    /*
     * Groupings
     */
    grouping authentication-container {
      description
        "A container defining authentication attributes.";
      container authentication {
        description
          "A container defining authentication attributes.";
        choice authentication-type {
          case key-chain {
            if-feature peer-key-chain;
            leaf key-chain {
              type key-chain:key-chain-ref;
              description
                "Reference to a key-chain.";
            }
          }
          case password {
            if-feature peer-key-chain;
            leaf key {
              type string;
              description
                "This leaf describes the authentication key.";
            }
            uses key-chain:crypto-algorithm-types;
          }
          description
            "Choice of authentication.";
        }
      }
    } // authentication-container

    grouping connect-source {
      description "Attribute to configure connect-source.";
      leaf connect-source {
        type if:interface-ref;
        must "/if:interfaces/if:interface[if:name = current()]/"
          + "ip:ipv4" {
          description
            "The interface must have IPv4 enabled.";
        }
        description
          "The interface is to be the source for the TCP connection.
```

```
            It is a reference to an entry in the global interface
            list.";
        }
      } // connect-source

      grouping global-config-attributes {
        description "Global MSDP configuration.";

        uses connect-source {
          if-feature global-connect-source;
        }
        container default-peer {
          if-feature global-default-peer;
          presence "";
          description
            "The default peer accepts all MSDP SA messages.
             A default peer is needed in topologies where MSDP peers do
             not coexist with BGP peers. The reverse path forwarding
             (RPF) check on SA messages can fail, and no SA messages are
             accepted. In these cases, you can configure the peer as a
             default peer and bypass RPF checks.";
          leaf peer-addr {
            type leafref {
              path "../../../peers/peer/address";
            }
            mandatory true;
            description
              "Reference to a peer that is in the peer list.";
          }
          leaf prefix-policy {
            if-feature global-default-peer-policy;
            type string;
            description
              "If specified, only those SA entries whose RP is permitted
               in the prefix list are allowed;
               if not specified, all SA messages from the default peer
               are accepted.";
          }
        } // default-peer

        container originating-rp {
          description
            "The container of originating-rp.";
          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.
             IP address of the interface is used in the RP field of an
             SA message entry. When Anycast RPs are used, all RPs use
             the same IP address. This parameter can be used to define
             a unique IP address for the RP of each MSDP peer.
             By default, the software uses the RP address of the
             local system.";
        }
      } // originating-rp

      uses sa-filter-container {
        if-feature global-sa-filter;
      }
      leaf sa-limit {
        if-feature global-sa-limit;
        type uint32;
        description
          "A limit on the number of SA entries accepted. By default,
           there is no limit.";
      }
      uses ttl-threshold {
        if-feature global-ttl-threshold;
      }
    } // global-config-attributes

    grouping global-state-attributes {
      description "Global MSDP state attributes.";
    } // global-state-attributes

    grouping peer-config-attributes {
      description "Per peer configuration for MSDP.";

      uses authentication-container;
      leaf enable {
        if-feature peer-admin-enable;
        type boolean;
        description
          "true to enable peer;
           false to disable peer.";
      }
      uses connect-source {
        if-feature peer-connect-source;
      }
      leaf description {
```

```
          if-feature peer-description;
          type string;
          description
            "The peer description.";
        }
        leaf mesh-group {
          type string;
          description
            "Configure this peer to be a member of a mesh group";
        }
        leaf peer-as {
          if-feature peer-as;
          type string;
          description
            "Peer's autonomous system number (ASN).";
        }
        uses sa-filter-container;
        leaf sa-limit {
          if-feature peer-sa-limit;
          type uint32;
          description
            "A limit on the number of SA entries accepted from this peer.
             By default, there is no limit.";
        }
        container timer {
          description "Timer attributes.";
          leaf connect-retry-interval {
            if-feature peer-timer-connect-retry;
            type uint16;
            units seconds;
            default 30;
            description "SHOULD be set to 30 seconds. ";
          }
          leaf holdtime-interval {
            if-feature peer-timer-holdtime;
            type uint16;
            units seconds;
            must ". > 3";
            default 75;
            description "The SA-Hold-Down-Period of this msdp peer.";
          }
          leaf keepalive-interval {
            if-feature peer-timer-keepalive;
            type uint16;
            units seconds;
            must ". > 1 and . < ../holdtime-interval";
            default 60;
            description "The keepalive timer of this msdp peer.";
```

```
        }
      } // timer
      uses ttl-threshold;
    } // peer-config-attributes

    grouping peer-state-attributes {
      description "Per peer state attributes for MSDP.";

      leaf session-state {
        type enumeration {
          enum disabled  {
            description "Disabled.";
          }
          enum inactive {
            description "Inactive.";
          }
          enum listen {
            description "Listen.";
          }
          enum connecting {
            description "Connecting.";
          }
          enum established {
            description "Established.";
          }
        }
        description
          "Peer session state.";
        reference
          "RFC3618: Multicast Source Discovery Protocol (MSDP).";
      }
      leaf elapsed-time {
        type uint32;
        units seconds;
        description "Elapsed time for being in a state.";
      }
      leaf connect-retry-expire {
        type uint32;
        units seconds;
        description "Connect retry expire time of peer connection.";
      }
      leaf hold-expire {
        type uint32;
        units seconds;
        description "Hold expire time of peer connection.";
      }
      leaf is-default-peer {
        type boolean;
```

```
        description "If this peer is default peer.";
      }
      leaf keepalive-expire {
        type uint32;
        units seconds;
        description "Keepalive expire time of this peer.";
      }
      leaf reset-count {
        type uint32;
        description "The reset count of this peer.";
      }
      uses statistics-container;
    } // peer-config-attributes

    grouping sa-cache-state-attributes {
      description "SA cache state attributes for MSDP.";

      leaf up-time {
        type uint32;
        units seconds;
        description "The up time of this sa cache.";
      }
      leaf expire {
        type uint32;
        units seconds;
        description "If this cache has expired.";
      }
      leaf holddown-interval {
        type uint32;
        units seconds;
        description "Holddown timer value for SA forwarding.";
      }
      leaf peer-learned-from {
        type inet:ipv4-address;
        description
          "The address of peer that we learned this SA from .";
      }
      leaf rpf-peer {
        type inet:ipv4-address;
        description "RPF peer.";
      }
    } // sa-cache-state-attributes

    grouping sa-filter-container {
      description "A container defining SA filters.";
      container sa-filter {
        description
          "Specifies an access control list (ACL) to filter source
```

```
            active (SA) messages coming in to or going out of the
            peer.";
          leaf in {
            type string;
            description
              "Filters incoming SA messages only.";
          }
          leaf out {
            type string;
            description
              "Filters outgoing SA messages only.";
          }
        } // sa-filter
      } // sa-filter-container

      grouping ttl-threshold {
        description "Attribute to configure TTL threshold.";
        leaf ttl-threshold {
          type uint8 {
            range 1..255;
          }
          description
            "Maximum number of hops data packets can traverse before
             being dropped.";
        }
      } // sa-ttl-threshold

      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.";
      leaf rpf-failure {
        type uint32;
        description "";
      }
    } // statistics-error

    grouping statistics-queue {
      description
        "A grouping defining queue statistics
        attributes.";
      leaf size-in {
        type uint32;
        description
          "The size of the input queue.";
      }
      leaf size-out {
        type uint32;
        description
          "The size of the output queue.";
      }
    } // statistics-queue

    grouping statistics-sent-received {
      description
        "A grouping defining sent and received statistics
        attributes.";
      leaf keepalive {
        type yang:counter64;
        description
          "The number of keepalive messages.";
      }
```

```
        leaf notification {
          type yang:counter64;
          description
            "The number of notification messages.";
        }
        leaf sa-message {
          type yang:counter64;
          description
            "The number of SA messages.";
        }
        leaf sa-response {
          type yang:counter64;
          description
            "The number of SA response messages.";
        }
        leaf sa-request {
          type yang:counter64;
          description
            "The number of SA request messages.";
        }
        leaf total {
          type yang:counter64;
          description
            "The number of total messages.";
        }
      } // statistics-sent-received

      /*
       * Configuration data nodes
       */
      augment "/rt:routing/rt:control-plane-protocols" {
        description
          "MSDP augmentation to routing instance configuration.";

        container msdp {
          presence "Container for MSDP protocol.";
          description
            "MSDP configuration data.";

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

          container peers {
            description
              "Containing a list of peers.";
```

```
          list peer {
            key "address";
            description
              "List of MSDP peers.";
            leaf address {
              type inet:ipv4-address;
              description
                "";
            }
            uses peer-config-attributes;
          } // peer
        } // peers
      } // msdp
    } // augment

    /*
     * Operational state data nodes
     */
    augment "/rt:routing-state/rt:control-plane-protocols" {
      description
        "MSDP augmentation to routing instance state.";

      container msdp {
        presence "Container for MSDP protocol.";
        description
          "MSDP state data.";

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

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

          list peer {
            key "address";
            description
              "List of MSDP peers.";
            leaf address {
              type inet:ipv4-address;
              description
                "The address of peer";
            }
            uses peer-config-attributes;
```

```
            uses peer-state-attributes;
          } // peer
        } // peers

        container sa-cache {
          description
            "The sa cache information.";
          list entry {
            key "group source-addr";
            description "";
            leaf group {
              type inet:ipv4-address;
              description "The group address of this sa cache.";
            }
            leaf source-addr {
              type union {
                type enumeration {
                  enum '*' {
                    description "The source addr of this sa cache.";
                  }
                }
                type inet:ipv4-address;
              }
              description "";
            }
            list origin-rp {
              key "rp-address";
              description
                "";
              leaf rp-address {
                type inet:ip-address;
                description "The rp address.";
              }
              leaf is-local-rp {
                type boolean;
                description "";
              }
              leaf sa-adv-expire {
                type uint32;
                units seconds;
                description
                  "Periodic SA advertisement timer exipiring time on
                   a local RP.";
              }
            }
            uses sa-cache-state-attributes;
          } // entry
        } // sa-cache
```

```
      } // msdp
    } // augment

    /*
     * RPCs
     */
    rpc msdp-clear-peer {
      description
        "Clears the session to the peer.";
      input {
        leaf peer-address {
          type inet:ipv4-address;
          description
            "Address of peer to be cleared. If this is not provided
             then all peers are cleared.";
        }
      }
    }

    rpc msdp-clear-sa-cache {
      if-feature rpc-clear-sa-cache;
      description
        "Clears MSDP source active (SA) cache entries.";
      input {
        container entry {
          presence "";
          description
            "The SA cache (S,G) or (*,G) entry to be cleared. If this
             is not provided, all entries are cleared.";
          leaf group {
            type inet:ipv4-address;
            mandatory true;
            description "";
          }
          leaf source-addr {
            type union {
              type enumeration {
                enum '*' {
                  description "";
                }
              }
              type inet:ipv4-address;
            }
            description "";
          }
        } // s-g
        leaf peer-address {
          type inet:ipv4-address;
```

```
            description
              "Peer IP address from which MSDP SA cache entries have been
               learned. If this is not provided, entries learned from all
               peers are cleared.";
          }
          leaf peer-as {
            type string;
            description
              "ASN from which MSDP SA cache entries have been learned.
               If this is not provided, entries learned from all AS's
               are cleared.";
          }
        }
      }
    }
    <CODE ENDS>
```

8. Contributors

The authors would like to thank Yisong Liu (liuyisong@huawei.com), Benchong Xu (xu.benchong@zte.com.cn), Tanmoy Kundu (tanmoy.kundu@alcatel-lucent.com) for their valuable contributions.

9. Normative References

[I-D.ietf-netmod-routing-cfg] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-24 (work in progress), October 2016.

[RFC3618] Fenner, B., Ed. and D. Meyer, Ed., "Multicast Source Discovery Protocol (MSDP)", RFC 3618, DOI 10.17487/RFC3618, October 2003, <http://www.rfc-editor.org/info/rfc3618>.

[RFC4624] Fenner, B. and D. Thaler, "Multicast Source Discovery Protocol (MSDP) MIB", RFC 4624, DOI 10.17487/RFC4624, October 2006, <http://www.rfc-editor.org/info/rfc4624>.

[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>.

Authors' Addresses

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

Email: xliu@kuatrotech.com

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

Email: zhang.zheng@zte.com.cn

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

Email: anishp@juniper.net

Mahesh Sivakumar
Cisco Systems
510 McCarthy Boulevard
Milpitas, California
USA

Email: masivaku@cisco.com

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

Email: guofeng@huawei.com

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

Email: pete.mcallister@metaswitch.com