PIM Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

Hermin Anggawijaya
Allied Telesis Labs, NZ

October 19, 2015

Improving IGMPv3 and MLDv2 Resilience
draft-anggawijaya-pim-resilient-gmp-00

Abstract

Host devices send IGMP or MLD group membership report messages to tell the designated router (DR) which multicast groups they want to receive.
These messages are sent repeatedly up to maximum number of times determined by the 'Robustness Variable' setting. However, in certain situations, those messages could get lost.
This draft proposes a mechanism whereby host devices attached to a local area network where the querier and the DR are the same device, can request the querier to acknowledge each report message it receives, ensuring report messages reception by the DR.

Status of this Memo

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

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/1id-abstracts.html

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

This Internet-Draft will expire in February, 2016.

Copyright Notice

Table of Contents

1. Introduction

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

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

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

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

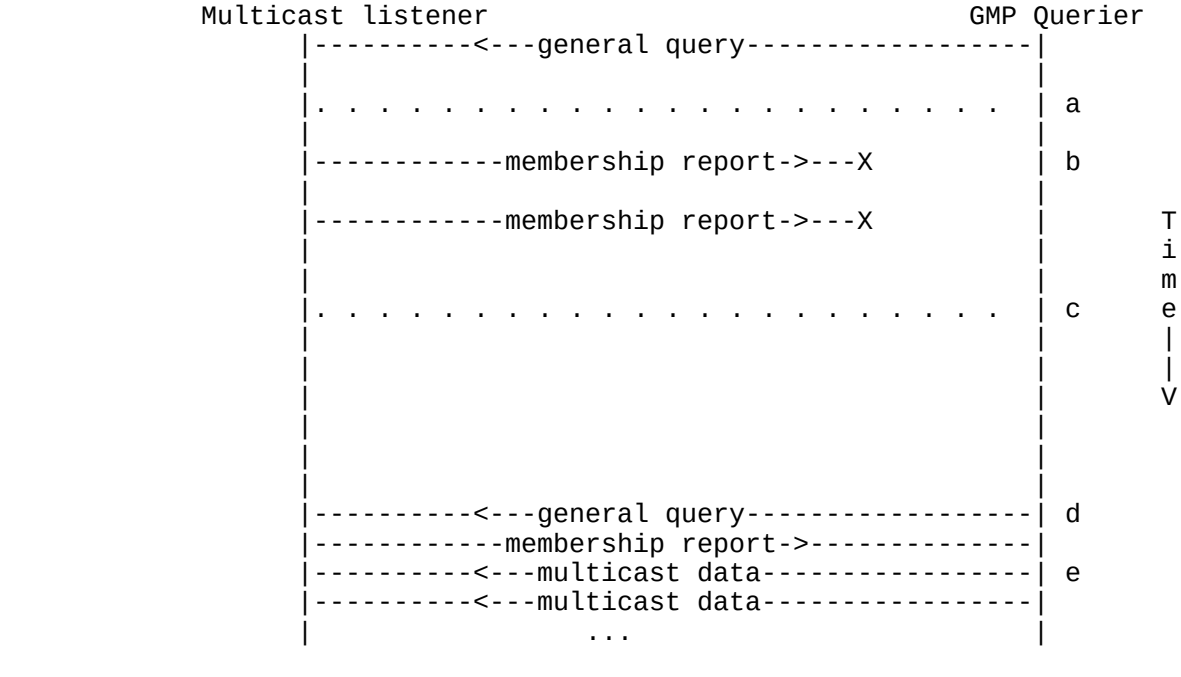
Early physical and link layer technologies such as 802.3 10Base-T, are characterised by the possibility of frames getting lost during transmission. To compensate for the possibility of frame losses, both the hosts and querier are required to repeat their GMP protocol message transmissions. The number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

Although wired link layer technology improvements have been made to minimise the possibility of frame loss, the following recent networking trends have tended to negate these effects:
The growing tendency to increase the number of devices operating within a single broadcast domain, then interconnecting multiple domains using either bridged (or layer two switched) links.

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

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

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



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

Diagram 1. Current GMP operation with transient state on querier

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

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

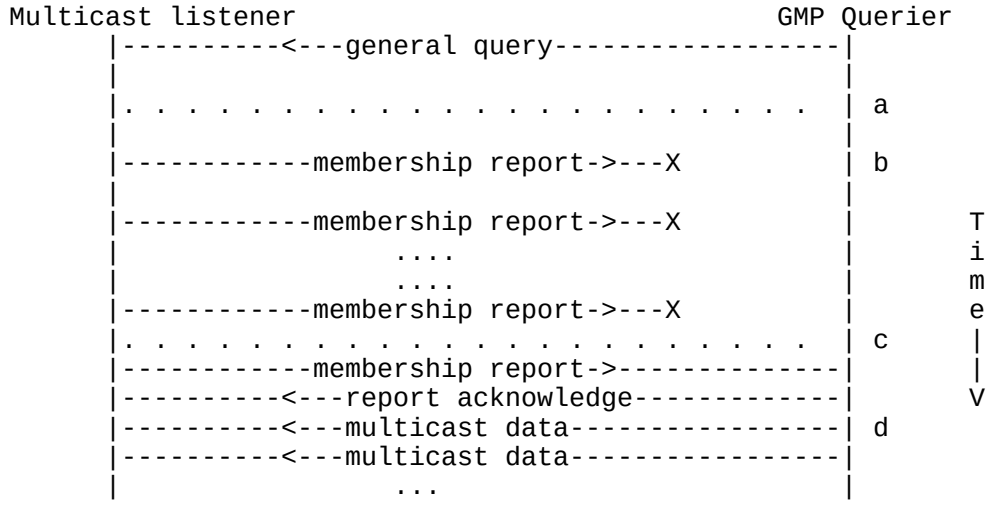
2. Terminology

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

3. Resilient GMP

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

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



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

Diagram 2. Resilient GMP operation with transient state on querier

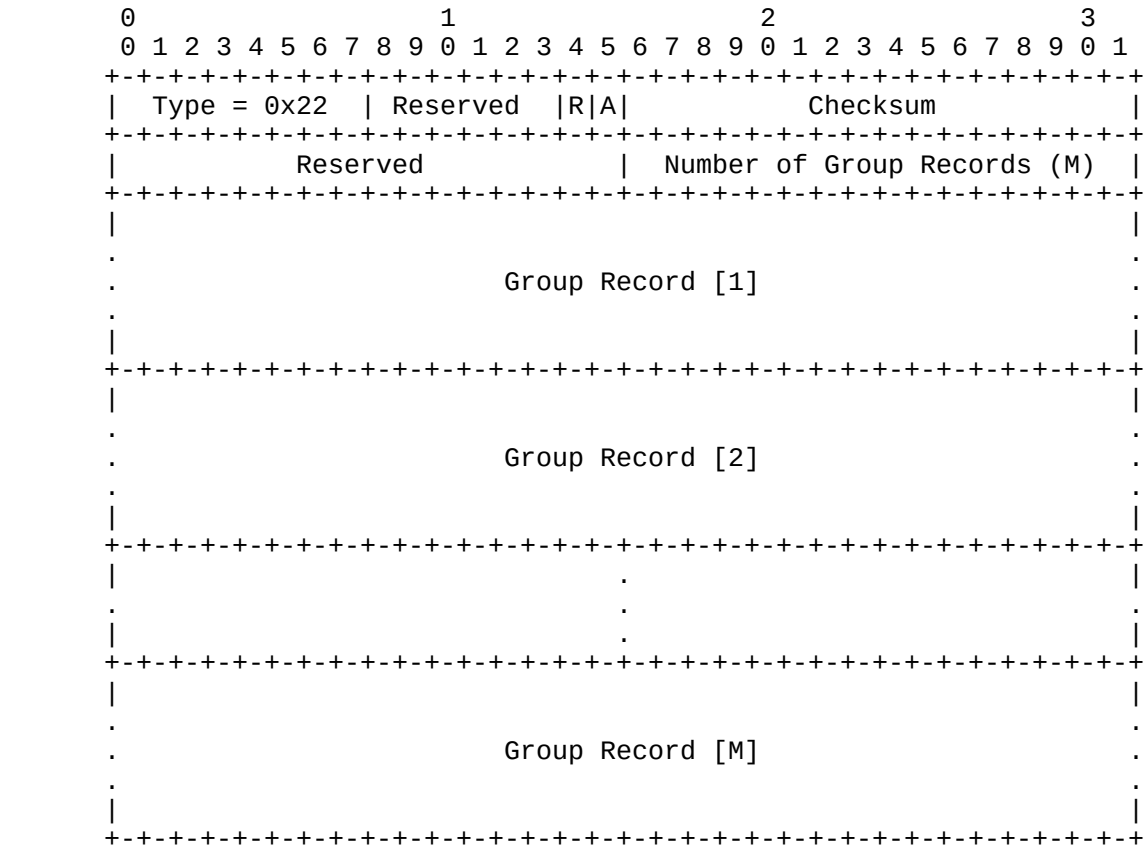
In order to apply this mechanism, a new message format is required that includes a flag to enable multicast listeners to request an acknowledgement from the querier. This message format is defined in the next section.

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

3.1. Resilient IGMP

3.1.1. Message Formats

IGMPv3 Membership Report



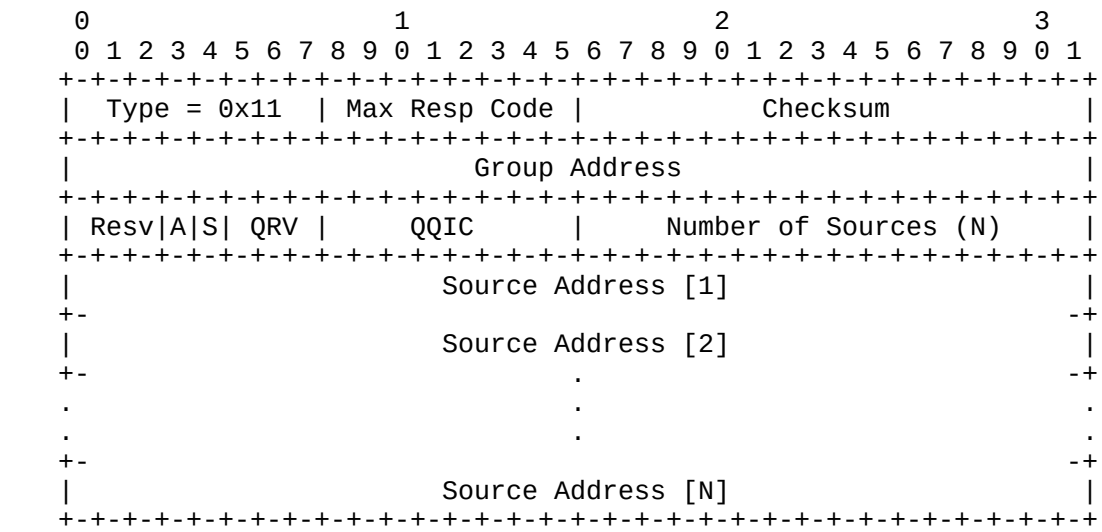
Please see [RFC3376] for the definitions of the existing fields.

New flags defined in the membership report message are:

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

A: 'Membership report acknowledgement' flag. Setting this flag indicates that this message is an acknowledgement to the membership report message contained in this message.

IGMPv3 Query



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

The new flag defined in the IGMPv3 Query is:

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

3.1.2. Host Behaviour

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

1. By default, an implementation of IGMP adhering to this draft MUST enable the new listener behaviour as specified below. And if configured to do so, it MAY also fall back to operate as per [RFC3376] specification. This provision also ensures that the an existing IGMPv3 implementation can operate with a querier implementing this draft.

2. A multicast listener adhering to this new specification MUST send its group membership reports with the R-flag set, and A-bit clear, in order to indicate its requirement that the querier acknowledges its group membership reports

3. If a multicast listener sends a group membership report message with an unspecified source IP address, it MUST not send the reports with the R-flag set.

4. As soon as a multicast listener receives a general query with the A-flag cleared, it MUST fall back to normal IGMPv3 operation as per [RFC3376] specification, i.e. all its membership reports must be sent with both R-flag and A-flag cleared, and it can only retransmit its messages (RV-1) number of times. This ensures that the new implementation of IGMP listener can inter-operate with existing IGMP querier implementations.

5. The multicast listener MUST keep retransmitting its membership report until:
   a. It receives an acknowledgement message from the querier or
   b. It detects that there are no querier present on the link. The listener will assume this situation if it has received no queries for a 'Querier Live Time' period since it transmitted the original membership report message. The Querier Live Time is calculated as 2.5 times the Query Interval time.

6. The retransmission of the group membership reports MUST be done at random intervals within the range (0, [Unsolicited Report Interval]).

7. If a multicast listener's interface state changes, the retransmission of the previous membership report messages MUST stop, because the state machine will send new membership report messages reflecting the state change.

8. A multicast listener MUST process only valid acknowledgement messages. Invalid messages MUST be dropped silently. A valid acknowledgement message MUST obey the following rules:
   a. The destination IP address must be equal to the IP address of the receiving interface, and cannot be a multicast, broadcast nor unspecified IP address.
   b. The R-flag of the report must be cleared and A-flag must be set.
   c. The checksum must be correct.
   d. The listener is able to match the acknowledgement message with the member report message waiting for an acknowledgement.

9. If a multicast listener send multicast membership reports consisting only of group records indicating that the host no longer wish to receive multicast data for the groups specified, such as BLOCK_OLD_SOURCES record, the report MAY be sent without the R-bit set. Note that by doing this, if the report is dropped then the DR will not send multicast leave to its upstream router and may result in unnecessary multicast data forwarding.

3.1.3. Router Behaviour

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

1. By default, a querier adhering to this draft MUST send queries with the A-flag set. The implementation SHOULD also enable operators to turn the new behaviour off administratively.

2. If a querier receives a membership report message it MUST perform the same report message validation as specified in [RFC3376], i.e. that all invalid report messages MUST be silently discarded. And in addition, it MUST also perform the following checks:
   a. That the A-flag is clear
   b. If the R-flag is set, check that the source IP address is not the unspecified IP address (0.0.0.0).

3. If a querier receives a valid membership report message with the R-flag set, it MUST send an acknowledgement by retransmitting the same report message back to the listener after performing the following alterations to the message:
   a. The source IP address MUST be set to the that of the receiving interface.
   b. The destination IP address MUST be set to the source IP address contained in the original message.
   c. The R-flag is cleared, and the A-flag is set.
   d. The checksum of the packet is recalculated.

4. If a querier receives a valid membership report message with the R-flag cleared, then it will just process the report message as per [RFC3376] specification.

5. When a querier sends a query, it MUST set the A flag, unless it is configured not to do so by the administrator.

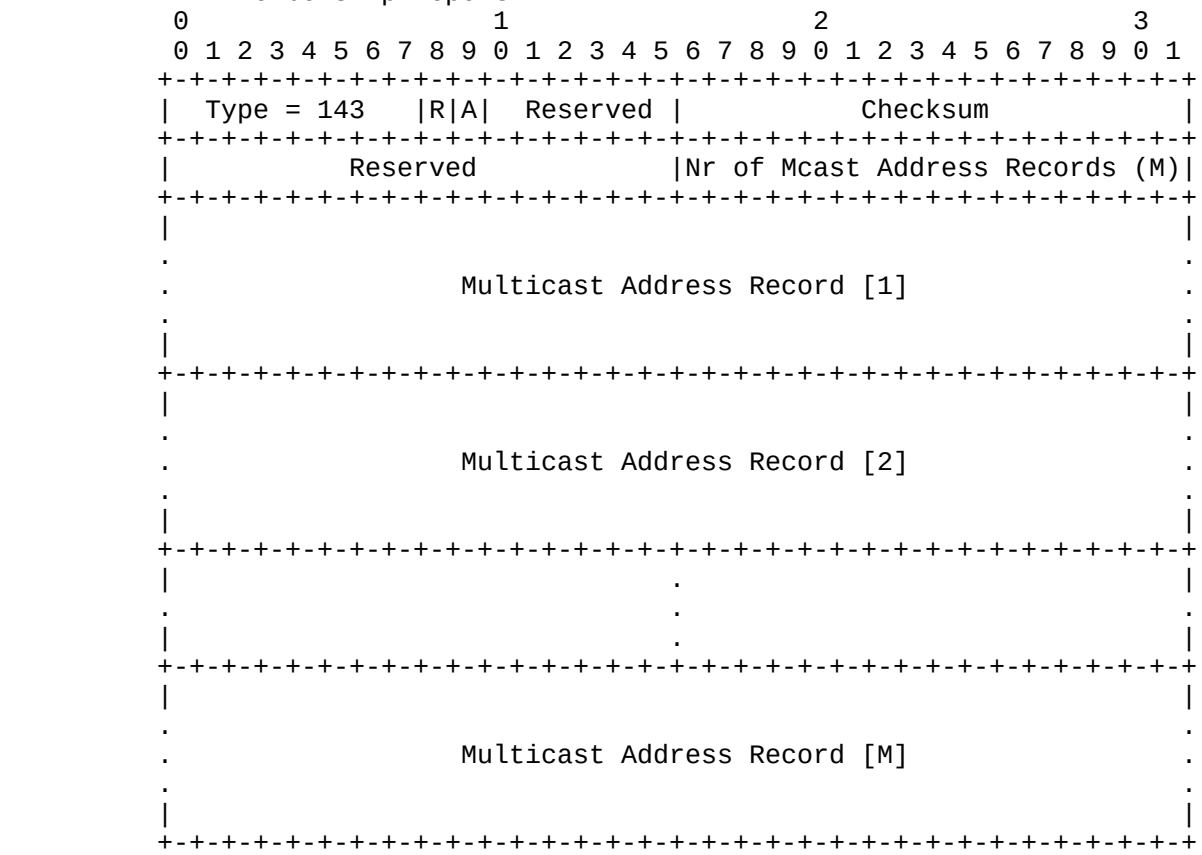
3.1.4. Backward Compatibility

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

3.2. Resilient MLD

3.2.1. Message Formats

MLDv2 Membership Report



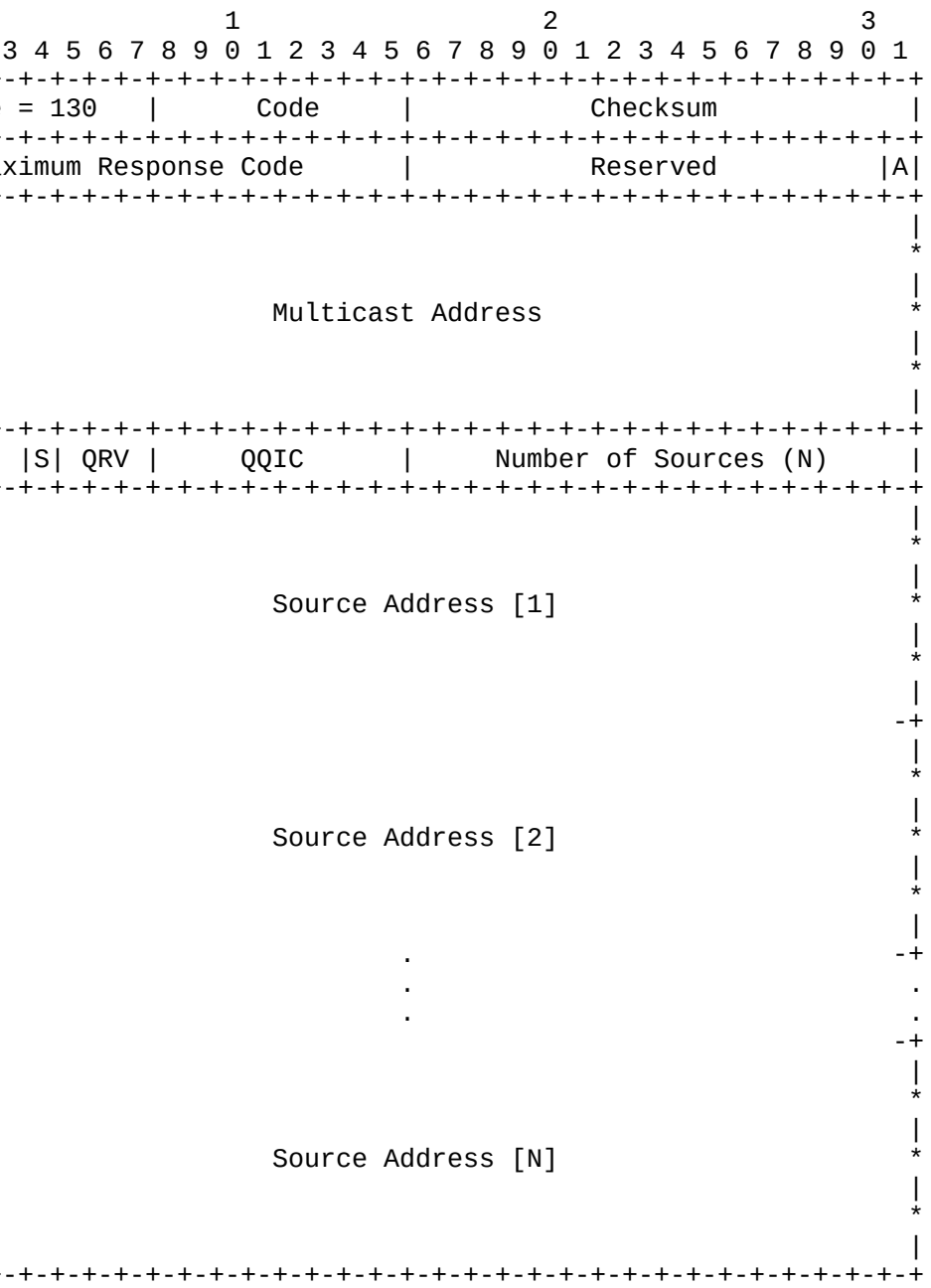
Please see [RFC3810] for the definitions of the existing fields.

The new flags defined in the membership report message are:

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

A: 'Membership report acknowledgement' flag. Setting this flag indicates that this message is an acknowledgement to the membership report message contained in this message.

MLDv2 Query



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

The new flag defined in MLDv2 Query is:
A : 'Acknowledgement Capable Querier' flag. Setting this flag indicates that this querier is capable of sending an acknowledgement to membership reports toward multicast listeners.

3.2.2. Host Behaviour

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

1. By default, an implementation of MLD adhering to this draft MUST enable the new listener behaviour as specified below. And if configured to do so, it MAY also fall back to operate as per [RFC3810] specification. This also ensures that existing MLDv2 implementations can inter-operate with queriers implementing this draft.

2. A multicast listener adhering to this new specification MUST send a group membership reports with the R-flag set, and the A-bit clear, indicating its wish that the querier acknowledges the listener's group membership reports.

3. If a multicast listener sends a membership report message with an unspecified source IPv6 address (::), it MUST not send the reports with the R-flag set.

4. As soon as a multicast listener receives a general query with the A-flag cleared, it MUST fall back to normal MLDv2 operation as per [RFC3810] specification, i.e. all its membership reports must be sent with both R-flag and A-flag cleared, and it can only retransmit its messages (RV-1) number of times. This ensures that the new implementation of MLD listener can inter-operate with existing MLD querier implementations.

5. The multicast listener MUST keep retransmitting the membership report until:
   a. It receives an acknowledgement message from the querier or
   b. It detects that there are no querier present on the link. The listener will assume this situation if it has receives no queries for 'Querier Live Time' period since it transmitted the original membership report message. The Querier Live Time is calculated as 2.5 times the Query Interval time.

6. The retransmission of the group membership reports MUST be done at random intervals within the range (0, [Unsolicited Report Interval]).

7. If a multicast listener's interface state changes, retransmission of the previous membership report message MUST stop, because the

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

8. A multicast listener MUST process only valid acknowledgement messages. Invalid messages MUST be dropped silently. A valid acknowledgement message MUST obey the following rules:
   a. The destination IPv6 address must be equal to that of the receiving interface, and MUST NOT be a multicast or the unspecified (::) IPv6 address.
   b. The R-flag of the report must be cleared and A-flag must be set.
   c. The checksum must be correct.
   d. The listener is able to match the acknowledgement message with the member report message waiting for an acknowledgement.

9. If a multicast listener send multicast membership reports consisting only of group records indicating that the host no longer wish to receive multicast data for the groups specified, such as BLOCK_OLD_SOURCES record, the report MAY be sent without the R-bit set. Note that by doing this, if the report is dropped then the DR will not send multicast leave to its upstream router and may result in unnecessary multicast data forwarding.

3.2.3. Router Behaviour

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

1. By default, querier implementation adhering to this draft MUST send queries with the A-flag set. The implementation SHOULD also enable operators to turn the new behaviour off administratively.

2. If a querier receives a membership report message it MUST perform the same report message validation as specified in [RFC3810]. All invalid report messages MUST be silently discarded. It MUST also do the following checks:
   a. Ensure that the A-flag is clear.
   b. Check that if the R-flag is set, the source IPv6 address is not the unspecified IPv6 address (::).

3. If a querier receives a valid membership report message with the R-flag set, it MUST send an acknowledgement by retransmitting the same report message back to the listener after performing the following mandatory alterations to the message:
   a. Setting the source IPv6 address to the IPv6 address of the receiving interface.
   b. Setting the destination IPv6 address to the original source IPv6 address of the message.
   c. Clear the R-flag, and set the A-flag.
   d. Recalculate the packet’s checksum.

4. If a querier receives a valid membership report message with the R-flag unset, then it will process the report message as per [RFC3810] specification.

5. When a querier sends a query, it MUST set the A-flag, unless it is configured not to do so by the administrator.

3.2.4. Backward Compatibility

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

3.3 Operational Consideration

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

4. IANA Considerations

This document has no actions for IANA.

5. Security Considerations

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

5.1. On-link Query Message Forgery

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

5.2. On-link Membership Report Message Forgery

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

6. Acknowledgements

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

7. References

7.1. Normative References

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

[RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[RFC3810] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

7.2. Informative References

[VYNCKE] E. Vyncke , P. Thubert , E. Levy-Abegnoli , A. Yourtchenko, "Why Network-Layer Multicast is Not Always Efficient At Datalink Layer", draft-vyncke-6man-mcast-not-efficient, February 2014.

[MCBRIDE] M. McBride, C. Perkins, "Multicast Wifi Problem Statement", draft-mcbride-mboned-wifi-mcast-problem-statement, July, 2015.

Authors' Address

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

Email: hermin.anggawijaya@alliedtelesis.co.nz

PIM Working Group
Internet-Draft
Expires: January 7, 2016

H. Asaeda
NICT
E. Muramoto
Panasonic
LM. Contreras
Telefonica

July 6, 2015

Multiple Upstream Interface Support for IGMP/MLD Proxy
draft-asaeda-pim-multiif-igmpmldproxy-01

Abstract

This document describes the way of supporting multiple upstream interfaces for an IGMP/MLD proxy device. The proposed extension enables an IGMP/MLD proxy device to receive multicast sessions/ channels through the different upstream interfaces. The upstream interface is selected based on the subscriber address prefixes, channel/session IDs, and interface priority values. A mechanism for upstream interface takeover that enables to switch from an inactive upstream interface to an active upstream interface is also described.

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 January 7, 2016.

Copyright Notice

Table of Contents

1. Introduction

The Internet Group Management Protocol (IGMP) [2][4] for IPv4 and the Multicast Listener Discovery Protocol (MLD) [3][4] for IPv6 are the standard protocols for hosts to initiate joining or leaving of multicast sessions. A proxy device performing IGMP/MLD-based forwarding (as known as IGMP/MLD proxy) [5] maintains multicast membership information by IGMP/MLD protocols on the downstream interfaces and sends IGMP/MLD membership report messages via the upstream interface to the upstream multicast routers when the membership information changes (e.g., by receiving solicited/ unsolicited report messages). The proxy device forwards appropriate multicast packets received on its upstream interface to each downstream interface based on the downstream interface's subscriptions.

According to the specification of [5], an IGMP/MLD proxy has *a single* upstream interface and one or more downstream interfaces. The multicast forwarding tree must be manually configured by designating upstream and downstream interfaces on an IGMP/MLD proxy device, and the root of the tree is expected to be connected to a wider multicast infrastructure. An IGMP/MLD proxy device hence performs the router portion of the IGMP or MLD protocol on its downstream interfaces, and the host portion of IGMP/MLD on its upstream interface. The proxy device must not perform the router portion of IGMP/MLD on its upstream interface.

On the other hand, there is a scenario in which an IGMP/MLD proxy device enables multiple upstream interfaces and receives multicast packets through these interfaces. For example, a proxy device having more than one interface may want to access to different networks, such as a global network like the Internet and local-scope networks. Or, a proxy device having wired link (e.g., ethernet) and high-speed wireless link (e.g., WiMAX or LTE) may want to have the capability to connect to the Internet through both links. These proxy devices shall receive multicast packets from the different upstream interfaces and forward to the downstream interface(s). Several other scenarios and subsequent requirements for the support of multiple upstream interfaces on IGMP/MLD proxy are documented in [7].

This document describes the mechanism that enables an IGMP/MLD proxy device to receive multicast sessions/channels through the different upstream interfaces. The mechanism is configured with either "channel-based upstream selection" or "subscriber-based upstream selection", or both of them. By channel-based upstream selection, an IGMP/MLD proxy device selects one or multiple upstream interface(s) from the candidate upstream interfaces "per channel/session". By subscriber-based upstream selection, an IGMP/MLD proxy device selects one or multiple upstream interface(s) from the candidate upstream interfaces "per subscriber/receiver".

When a proxy device transmits an IGMP/MLD report message, it examines the source and multicast addresses in the IGMP/MLD records of the report message. It then transmits the appropriate IGMP/MLD report message(s) from the selected upstream interface(s). When a proxy device selects "one" upstream interface from the candidate upstream interfaces per session/channel, it enables "load balancing" per session/channel. When a proxy device selects "more than two" upstream interfaces from the candidate upstream interfaces per session/channel, it potentially receives duplicate (redundant) packets for the session/channel from the different upstream interfaces simultaneously and provides "robust data reception".

A mechanism for "upstream interface takeover" is also described in this document; when the selected upstream interface is going down or the state of the link attached to the upstream interface is inactive, one of the other active candidate upstream interfaces takes over the upstream interface (if configured). The potential timer values to switch from an inactive upstream interface to an active upstream interface from a list of candidate upstream interfaces are discussed in this document as well.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT","SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

In addition, the following terms are used in this document.

Selected upstream interface (or simply, upstream interface):
A proxy device's interface in the direction of the root of the multicast forwarding tree. A proxy device performs the host portion of IGMP/MLD on its upstream interfaces. An upstream interface is selected from a list of candidate upstream interfaces.

Default upstream interface:
A default upstream interface is the upstream interface for multicast sessions/channels for which a proxy device cannot choose other interfaces as the upstream interface. A default upstream interface is configured.

Active upstream interface:
An active upstream interface is the upstream interface that has been receiving packets for specific multicast sessions/channels during the pre-defined active interval.

Inactive upstream interface:
An inactive upstream interface is the interface that has not received packets for specific multicast sessions/channels during the pre-defined active interval.

Downstream interface:
Each of a proxy device's interfaces that is not in the direction of the root of the multicast forwarding tree. A proxy device performs the router portion of IGMP/MLD on its downstream interfaces.

Candidate upstream interface:
An interface that potentially becomes an upstream interface of the proxy device. A list of candidate upstream interfaces is configured

with subscriber address prefixes, channel/session IDs, and priority values on an IGMP/MLD proxy device.

Channel/session ID:
Channel/session ID consists of source address prefix and multicast address prefix for which a candidate upstream interface supposes to be an upstream interface for specified multicast sessions/channels. Both or either source address prefix and/or multicast address prefix can be "null".

3. Upstream Selection Mechanism

3.1. Channel-Based Upstream Selection

An IGMP/MLD proxy device selects an upstream interface one or multiple upstream interface(s) from the candidate upstream interfaces "per channel/session". This is done by "channel-based upstream selection". It enables a proxy device to use one or multiple upstream interface(s) per session/channel from the "candidate upstream interfaces" based on the "channel/session ID" configuration (as will be in Section 4).

3.2. Subscriber-Based Upstream Selection

An IGMP/MLD proxy device selects one or multiple upstream interface(s) from the candidate upstream interfaces "per subscriber/ receiver". This is done by "subscriber-based upstream selection". It enables a proxy device to use one or multiple upstream interface(s) per session/channel from the "candidate upstream interfaces" based on the "subscriber address prefix" configuration (as will be in Section 4).

3.3. Multiple Upstream Interface Selection for Robust Data Reception

When more than one candidate upstream interface is configured with the same source and multicast addresses for the "channel/session IDs", and "interface priority values" (as will be in Section 4.3) are identical, these candidate upstream interfaces act as the upstream interfaces for the sessions/channels and receive the packets simultaneously. This multiple upstream interface selection implements duplicate packet reception from redundant paths. It may improve data reception quality or robustness for a session/channel, as the same multicast data packets can come from different upstream interfaces simultaneously. However, this robust data reception does not guarantee that the packets come from disjoint paths. It only configures that the adjacent upstream routers are different.

4. Candidate Upstream Interface Configuration

Candidate upstream interfaces are the interfaces from which an IGMP/MLD proxy device selects as an upstream interface. The upstream interface selection works with the configurations of "subscriber address prefix", "channel/session ID", and "interface priority value".

4.1. Address Prefix Record

An IGMP/MLD proxy device can configure the "subscriber address prefix" and "channel/session ID" for each candidate upstream interface. Channel/session ID consists of "source address prefix" and "multicast address prefix". Subscriber address prefix and source address prefix MUST be a valid unicast address prefix, and multicast address prefix MUST be a valid multicast address prefix. A proxy selects an upstream interface from its candidate upstream interfaces based on the configuration of the following address prefix record:

(subscriber address prefix, (channel/session ID))

where channel/session ID includes:

(source address prefix, multicast address prefix)

The default values of these address prefixes are "null". Null source address prefix represents the wildcard source address prefix, which indicates any host. Null multicast address prefix represents the wildcard multicast address prefix, which indicates the entire multicast address range (i.e., ’224.0.0.0/4’ for IPv4 or ’ff00::/8’ for IPv6).

The candidate upstream interface having the configuration of subscriber address prefix is prioritized. If network operators want to assign a specific upstream interface for specific subscribers without depending on source and multicast address prefixes, both source and multicast addresses in the address prefix record is configured "null".

If network operators want to select specific upstream interface(s) without depending on subscriber address prefix, subscriber address prefix in the address prefix record is configured "null".

4.2. Channel/Session ID

Channel/session ID configuration consists of source and multicast address prefixes. Both/either source and/or multicast address may be configured "null". A candidate upstream interface having non-null

source and multicast address configuration is prioritized for the upstream interface selection. For example, if a proxy device has two candidate upstream interfaces for the same multicast address prefix and one of them has non-null source address configuration, then that candidate upstream interface is selected for the source and multicast address pair. The other candidate upstream interface is selected for the configured multicast address prefix except the source address configured by the prior interface.

Source address prefix configuration takes priority over multicast address prefix configuration. For example, consider the case that an IGMP/MLD proxy device has a configuration with source address prefix S_p for the candidate upstream interface A and multicast address prefix G_p for the candidate upstream interface B. When it deals with an IGMP/MLD record whose source address, let's say S, is in the range of S_p, and whose multicast address, let's say G, is in the range of G_p, the proxy device selects the candidate upstream interface A, which supports the source address prefix, as the upstream interface, and transmits the (S,G) record via the interface A.

The same address prefix may be configured on different candidate upstream interfaces. When the same address prefix is configured on different candidate upstream interfaces, an upstream interface for that address prefix is selected based on each interface priority value (as will be in Section 4.3).

4.3. Interface Priority

An IGMP/MLD proxy device can configure the "interface priority" value for each candidate upstream interface. It is an integer value and is part of the configuration. The default value of the interface priority is the lowest value.

The interface priority value effects only when either of the following conditions is satisfied.

- None of the candidate upstream interfaces configures both the subscriber address prefix and the channel/session ID.
- More than one candidate upstream interface configures the same channel/session IDs but does not configure the subscriber address prefix.

In these conditions, the candidate upstream interface with the highest priority is chosen as the upstream interface. And as stated in Section 3.3, if the priority values for candidate upstream interfaces are also identical, all of these interfaces act as the

upstream interfaces for the configured channel/session ID and may receive duplicate packets.

4.4. Default Upstream Interface

An IGMP/MLD proxy device SHOULD configure "a default upstream interface" for all incoming sessions/channels. A default upstream interface is selected as the upstream interface, when none of the candidate upstream interfaces configures subscriber address prefix, channel/session ID, or interface priority value, or with either of the following conditions.

- o None of the candidate upstream interfaces configures both the subscriber address prefix, the channel/session ID, and identical interface priority value.

- o More than one candidate upstream interface configures the same channel/session IDs and identical interface priority value, but does not configure the subscriber address prefix.

If a default upstream interface is not configured on an IGMP/MLD proxy device, the candidate upstream interface whose IPv4/v6 address is the highest of others is configured as the default upstream interface for the proxy device.

5. Upstream Interface Takeover

5.1. Proxy Behavior

If a selected upstream interface is going down or inactive, or an adjacent upstream router is not working, the upstream interface can be disabled and the other active upstream interface listed in the candidate upstream interfaces covering the same channel/session ID can act as a new upstream interface. It recursively examines the list of the candidate upstream interfaces (except the disabled interface) and decides a new upstream interface from them. If no active candidate upstream interfaces exist, the default upstream interface takes its role.

This function called "upstream interface takeover" is a default function for a proxy device that enables multiple upstream interface support. If a proxy device simultaneously uses more than two upstream interfaces per session/channel, and one or some of these upstream interface(s) is/are inactive, the proxy device acts either of the following behaviors based on the configuration; (1) it only uses the active upstream interface(s) and does not add (i.e., does not complement) other upstream interfaces, (2) it uses the active upstream interface(s) and another candidate upstream interface whose

priority is highest among the configured upstream interfaces, or (3) it uses the active upstream interface(s) and the default upstream interface.

The condition whether the upstream adjacent router is active or not can be decided by checking the link/interface condition on the proxy device or detected by monitoring IGMP/MLD Query or PIM [6] Hello message reception on the link. There are the cases that PIM is not running on the link or IGMP/MLD Query messages are not always transmitted by the upstream router (e.g., because of enabling the explicit tracking function [8]). Therefore, network operators MUST configure either; (1) the proxy device disables the upstream interface takeover, (2) the proxy device triggers upstream interface takeover by detecting no IGMP/MLD Query message within the active interval, or (3) the proxy device triggers upstream interface takeover by detecting no PIM Hello message within the active interval, for each candidate upstream interface.

Network operators may want to keep out of use for the inactive upstream interface(s). This causes, for example, when subscriber-based upstream selection is configured, according to their accounting policy (because the specific subscribers are planned to use the specific upstream interface and cannot receive packets from other upstream interfaces.) In that case, this upstream interface takeover must be disabled, and the proxy device keeps using that interface as the upstream interface for them (and waits for working the interface later again).

5.2. Active Interval

Active interval is a period, after which a proxy device recognizes that the selected upstream interface is inactive. Active interval for each candidate upstream interface SHOULD be configured. The active interval values are different in the situation whether the network operators want to trigger by either IGMP/MLD or PIM messages. The default active interval to detect an inactive upstream interface is around twice of IGMP/MLD General Query interval and PIM Hello interval. Further discussion [TBD].

6. Automatic Upstream Interface Configuration

If there is no static configurations for a candidate upstream interface, by monitoring IGMP/MLD messages a proxy device can automatically configure the upstream interface for specific multicast channels/sessions. This may be enabled by new IGMP/MLD messages. Further discussion [TBD].

7. IANA Considerations

This document has no actions for IANA.

8. Security Considerations

This document neither provides new functions nor modifies the standard functions defined in [2][3][4]; hence there is no additional security consideration provided for these protocols themselves. On the other hand, it may be possible to encounter DoS attacks to make the function for upstream interface takeover stop if attackers illegally sends IGMP/MLD Query or PIM Hello messages on a LAN within a shorter period (i.e., before expiring the active interval for the upstream interface). To bypass such threats, it is recommended to capture the source addresses of IGMP/MLD Query or PIM Hello message senders and check whether the addresses correspond to the correct adjacent upstream routers. Consideration [TBD].

9. References

9.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to indicate requirement levels", RFC 2119, March 1997.

[2] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[3] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

[4] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, February 2010.

[5] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.

[6] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

9.2. Informative References

[7] Contreras, LM. and CJ. Bernardos, "Requirements for the extension of the MLD proxy functionality to support multiple upstream interfaces", draft-contreras-pim-multiple-upstreams-reqs-00 (work-in-progress), March 2015.

[8] Asaeda, H., "IGMP/MLD-Based Explicit Membership Tracking Function for Multicast Routers", draft-ietf-pim-explicit-tracking-11 (work-in-progress), March 2015.

Authors' Addresses

Hitoshi Asaeda
National Institute of Information and Communications Technology
Network Research Headquarters
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: asaeda@nict.go.jp

Eiichi Muramoto
Panasonic Corporation
Advanced Research Division Intelligence Research Laboratory
600 Saedo-cho, Tsuzuki-ku
Yokohama, Kanagawa 224-8539
Japan

Email: muramoto.eiichi@jp.panasonic.com

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://people.tid.es/LuisM.Contreras/

PIM Working Group
Internet-Draft
Intended status: Experimental
Expires: April 21, 2016

H. Asaeda
NICT

October 19, 2015

IGMP/MLD-Based Explicit Membership Tracking Function for Multicast Routers
draft-ietf-pim-explicit-tracking-12

Abstract

This document describes the IGMP/MLD-based explicit membership tracking function for multicast routers and IGMP/MLD proxy devices supporting IGMPv3/MLDv2. The explicit membership tracking function contributes to saving network resources and shortening leave latency.

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 21, 2016.

Table of Contents

1. Introduction

The Internet Group Management Protocol (IGMP) version 3 [2] for IPv4 and the Multicast Listener Discovery Protocol (MLD) version 2 [3] for IPv6 are the standard protocols used by member hosts and multicast routers. Lightweight IGMPv3 and Lightweight MLDv2 (or LW-IGMPv3 and LW-MLDv2) [4] are subsets of the standard IGMPv3 and MLDv2.

When a host starts/finishes listening to particular multicast channels, it sends IGMP/MLD State-Change Report messages specifying the corresponding channel information as the join/leave request to its upstream router (i.e., an adjacent multicast router or IGMP/MLD proxy device [8]). The "unsolicited" report messages are sent only when the host joins/leaves the channels. Since IGMP/MLD are non-reliable protocols, unsolicited report messages may be lost or may not reach upstream routers. To alleviate this problem, unsolicited report messages are retransmitted a number of times according to the value of the [Robustness Variable] defined in [2][3].

In addition, a querier router periodically sends IGMP/MLD General Query messages every General Query timer interval (i.e. [Query Interval] value defined in [2][3]). Upon receiving the query messages, the member hosts reply with "solicited" report messages. Routers then keep their membership state information up to date. However, this approach still does not guarantee that the membership state is always perfectly synchronized. To minimize the possibility of having outdated membership information, routers may shorten the periodic General Query timer interval. Unfortunately, this increases the number of transmitted solicited report messages and induces

network congestion. And the greater the amount of network congestion, the greater the potential for IGMP/MLD report messages being lost and the membership state information being outdated in the router.

IGMPv3 [2], MLDv2 [3], and these lightweight protocols [4] can provide the ability to keep track of the downstream (adjacent) multicast membership state in multicast routers, yet the specifications are not clearly given. This document describes the "IGMP/MLD-based explicit member tracking function" for multicast routers and a way for routers to implement the function. By enabling this explicit tracking function, routers can keep track of the downstream multicast membership state.

This document specifies an experimental extension to IGMPv3/MLDv2. It makes some fundamental changes to how IGMPv3/MLDv2 works in that membership state does not require periodic updates, and it partly turns IGMPv3/MLDv2 into a hard-state protocol. Also, a mechanism called "specific query suppression" with a robust link state is a new concept. It does not make the router send any specific query message(s) and immediately leave the group or sources when the sole member has left according to its membership state information. It is likely that experiences from early implementations and deployments will lead to at least minor changes in the protocol. Once there is some deployment experience, making this a Standards Track protocol should be considered. Experiments using this protocol extension only require support by pairs of multicast routers on a LAN, and need not be constrained to isolated networks.

This document describes the risk of having wrong membership state as well. The explicit tracking function does not change the reliability of the message transmission. The list of tracked member hosts may be outdated in the router because of host departure from the network without sending State-Change Report messages or loss of such messages due to network congestion. This document guides for setting up appropriate values or mechanisms used with the explicit tracking function in routers.

The explicit tracking function potentially requires a large amount of memory so that routers keep all membership states. Particularly when a router needs to maintain a large number of member hosts, this resource requirement might be sensitive. As the security consideration, this document describes that operators may decide to disable this function when their routers have insufficient memory resources, despite the benefits.

The explicit tracking function does not change message formats used by IGMPv3 [2], MLDv2 [3], and their lightweight version protocols

[4]; nor does it change a multicast data sender's and receiver's behavior.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [1].

3. Membership State Information

A router enabling the explicit tracking function maintains the "membership state information". When a multicast router receives a Current-State or State-Change Report message, it creates or modifies this membership state information to maintain the membership state up to date.

The membership state information consists of the following information:

(S, G, number of receivers, (receiver records))

where "S" denotes source address, "G" denotes group or multicast address, and each receiver record is of the form:

(IGMP/MLD membership/listener report sender's address)

In the state information, each S and G indicates a single IPv4/IPv6 address. S is set to "All sources" for Any-Source Multicast (ASM) communication (i.e., (*,G) join reception). In order to simplify the implementation, Lightweight-IGMPv3/MLDv2 [4] do not keep the state of (S,G) joined with EXCLUDE filter mode; if a router receives an (S,G) join/leave request with EXCLUDE filter mode from the downstream hosts, the router translates the request to a (*,G) join state/leave request and records the state and the receivers' addresses in the maintained membership state information.

The membership state information must be identified properly even though a receiver (i.e., IGMP/MLD Report sender) sends the identical report messages multiple times. And the maintained membership state information will be flushed when the router reboots or restarts the multicast routing processes.

4. Specific Query Suppression

In accordance with [2] and [3], when a router receives the State-Change Report and needs to confirm whether any hosts are still interested in a channel or not, the router sends the corresponding

Group-Specific or Group-and-Source Specific Query messages as defined in Section 6.4.2 of [2] and Section 7.4.2 of [3]. The queries sent by actions defined in these sections need to be transmitted [Last Member Query Count] (LMQC) or [Last Listener Query Count] (LLQC) times, once every [Last Member Query Interval] (LMQI) or [Last Listener Query Interval] (LLQI), in order to confirm the sole member. (The default values for LMQI/LLQI defined in [2][3] are 1 second. The default values for LMQC/LLQC are the [Robustness Variable] value whose default value is 2.) All member hosts joining the identical channel then reply with their own states after acquiring these query messages. However, transmitting a large number of IGMP/MLD Report messages consumes network resources, and this may pose a particular problem especially when many hosts joining the identical channel send these reports simultaneously.

The explicit tracking function provides a mechanism called "specific query suppression". With the specific query suppression, regardless of the LMQC/LLQC values, if the router receives one or more replies from the downstream member(s), it SHOULD stop (i.e., cancel) retransmitting the specific query message(s) for the specified source and/or group. It reduces the number of Group-Specific or Group-and-Source Specific Query messages transmitted from a router, and in turn reduces the number of Current-State Report messages transmitted from member hosts. This contributes to saving network resources.

The specific query suppression MAY define an option called "robust link state". If an operator is confident that the link is stable and robust enough and thus the tracked membership state information is perfectly synchronized with the current (actual) member hosts, s/he can enable the specific query suppression with a robust link state. A router enabling the specific query suppression with a robust link state does not send any specific query message(s) and immediately leave the group or sources when the sole member has left according to its membership state information. The specific query suppression with a robust link state hence does not rely on LMQC/LLQC and LMQI/ LLQI values. This contributes to shortening leave latency described in Section 5. However, this behavior requires that the router perfectly tracks all member hosts. (See a risk of wrong membership expectation described in Section 6.)

Note that the default behavior of the router that supports the explicit tracking function SHOULD disable this specific query suppression, in order to avoid the risk caused by the wrong membership expectation or by the case in which multiple multicast routers exist on a LAN and the querier router is not the forwarder router. The former case is described in Section 6. For the latter case, when the querier suppresses the specific query message transmission, and expects that the State-Change Report sender is not

the sole member of the channel, it does not send the specific query. Then the routers (including the forwarder) on the same LAN do not receive a Current-State Report message from the corresponding member hosts. The forwarder in this case may prune the routing path, although there are other member hosts subscribing to the channel on the LAN.

5. Shortening Leave Latency

A router enabling the explicit tracking function can shorten leave latencies by tuning the following values; [Last Member Query Count] (LMQC), [Last Listener Query Count] (LLQC), [Last Member Query Interval] (LMQI), [Last Listener Query Interval] (LLQI), and [Robustness Variable] values.

The [Last Member Query Interval] (LMQI) and [Last Listener Query Interval] (LLQI) values defined in the standard specifications [2][3] specify the maximum time allowed for a member host to send a responding Report. The [Last Member Query Count] (LMQC) and [Last Listener Query Count] (LLQC) are the number of Group-Specific Queries or Group-and-Source Specific Queries sent before the router assumes there are no local members. The [Last Member Query Time] (LMQT) and [Last Listener Query Time] (LLQT) values are the total time the router should wait for a report after the Querier has sent the first query.

The default values for LMQI/LLQI defined in [2][3] are 1 second, yet, for a router enabling the explicit tracking function, the LMQI/LLQI may be set to 1 second or shorter. As well, the default values for LMQC/LLQC are the [Robustness Variable] value whose default value is 2, yet the LMQC/LLQC may be set to 1 for the router. Smaller LMQC/ LLQC values give shorter LMQT/LLQT, which shorten the leave latencies.

Furthermore, if operators are confident that their link is fairly robust (e.g., the [Robustness Variable] value is appropriately configured so that the chances of unsolicited messages being lost are sufficiently low), and if the querier router always acts as the forwarder router for all multicast channels in the LAN, they will set smaller LMQC/LLQC and shorter LMQI/LLQI (and hence shorter LMQT/LLQT) with the specific query suppression, or enable the specific query suppression with a robust link state (Section 4) for their routers.

Note that setting smaller LMQC/LLQC and shorter LMQI/LLQI values or adopting the specific query suppression with a robust link state poses the risk of wrong membership state described in Section 6. Operators setting these values or enabling that mechanism must recognize this tradeoff.

6. Risk of Wrong Membership State

There are possibilities that a router's membership expectation is inconsistent due to an outdated membership state. For example, (1) a router expects that more than one corresponding member host exists on its LAN, but in fact no member host exists for that multicast channel, or (2) a router expects that no corresponding member host exists on its LAN, but in fact one or more than one member host exists for that multicast channel.

The first case may occur in an environment where the sole member host departs the network without sending a State-Change Report message. The router later detects that there is no member host for the corresponding channels when it does not receive a Current-State Report within the timeout of the response for the periodic General Query (and then the group or source timers are expired). However, this situation prolongs leave latency and wastes network resources since the router forwards unneeded traffic for a while.

The second case occurs when a router sends a specific query but does not receive a Current-State Report from a downstream host within an LMQT or LLQT period. It recognizes that no member host exists on the LAN and might prune the routing path. The router reestablishes the routing path when it receives the solicited report message for the channels. However, the downstream hosts may loose the data packets until the routing path is reestablished and the data forwarding is restarted.

If operators do not believe that their link is fairly robust or that they can configure the [Robustness Variable] value appropriately, they may configure the LMQC/LLQC value to 2 (the default value of the [Robustness Variable] value) or bigger value for their routers. In this case, the routers would enable the explicit tracking function but may want to disable the specific query suppression specified in Section 4. Such configurations will not contribute to saving network resources, but reduce the risk of the incorrect membership expectation.

7. All-Zero and Unspecified Source Addresses

The IGMPv3 specification [2] mentions that an IGMPv3 report is usually sent with a valid IP source address, yet it permits a host to use the 0.0.0.0 source address (since the host has not yet acquired an IP address), and routers must accept a report with this source address.

When a router enabling the explicit tracking function receives IGMP report messages with an all-zero source address, it deals with the

IGMP report messages correctly as defined in [2] and continuously keeps track of the membership state. However, the router SHOULD NOT maintain the host specifying all-zero source address in its membership state information. The router will maintain its membership state information by checking Current-State reports as ordinary routers do.

On the other hand, the MLDv2 specification [3] mentions that routers silently discard a message that is sent with an invalid link-local address or sent with the unspecified address (::), without taking any action, because of security considerations. According to this specification, whether the explicit tracking function is used or not, a router does not deal with a member hosts sending an MLD report message with the unspecified source address.

8. Compatibility with Older Version Protocols

The explicit tracking function does not work with older versions of IGMP or MLD, IGMPv1 [5], IGMPv2 [6], or MLDv1 [7], because a member host using these protocols enables "membership report suppression" by which the host will cancel sending pending membership reports if a similar report is observed from another member on the network.

To preserve compatibility with older versions of IGMP/MLD, routers supporting IGMPv3/MLDv2 enable the host compatibility mode defined in [2][3]. The host compatibility mode of an interface changes the operational protocol version on the LAN whenever an older version query (than the current compatibility mode) is heard or when certain timer conditions occur. The routers can hence support downstream hosts that are not upgraded to the latest versions and run membership report suppression.

Therefore, if a multicast router supporting IGMPv3/MLDv2 and enabling the explicit tracking function changes its compatibility mode to the older versions, the router SHOULD disable the explicit tracking function while it acts as the older version router.

9. Interoperability

There might be various ways to implement the explicit tracking function. Some existing implementations may not implement the mechanisms such as specific query suppression described in this document. Yet, the explicit tracking function does not change on-wire behavior, and the function or mechanisms described in this document do not break the interoperability between the existing implementations and the implementation based on this specification.

On the other hand, for the future implementation for the explicit tracking function, since this document specifies the minimum but effective sets of the explicit tracking function, it is RECOMMENDED to refer and follow this specification as the standard implementation for that function.

10. IANA Considerations

This document has no actions for IANA.

11. Security Considerations

The explicit tracking function potentially requires a large amount of memory so that routers keep all membership states. It gives some impact in the cases where (1) a router attaches to a link or an IGMP/MLD proxy device [8] that has a large number of member hosts, and a router has insufficient memory resources to maintain a large number of member hosts, or (2) a malicious host sends a large number of invalid IGMP/MLD State-Change Report messages without any intent to join the specified channels.

For the first case, operators may disable the explicit tracking function, despite the benefits mentioned above. For the second case, some serious threats may be induced. For instance;

1. Transmitting a large number of invalid IGMP/MLD report messages consumes network resources.

2. Keeping a large number of invalid membership states on a router consumes the router’s memory resources.

3. Dealing with a large number of invalid membership states on a router consumes the router’s CPU resources.

In order to mitigate such threats, a router enabling the explicit tracking function may limit a total amount of membership information the router can store, or may rate-limit State-Change Report messages per host. When the router enables rate-limiting per host, the router MAY ignore the received State-Change Report messages to minimize the processing overhead or prevent DoS attacks. The rate limit is left to the router’s implementation.

12. Acknowledgements

Luis M. Contreras, Toerless Eckert, Adrian Farrel, Sergio Figueiredo, Bharat Joshi, Nicolai Leymann, Magnus Nystrom, Stig Venaas, and others provided many constructive and insightful comments.

13. References

13.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to indicate requirement levels", RFC 2119, March 1997.

[2] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, October 2002.

[3] Vida, R. and L. Costa, "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, June 2004.

[4] Liu, H., Cao, W., and H. Asaeda, "Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols", RFC 5790, February 2010.

13.2. Informative References

[5] Deering, S., "Host Extensions for IP Multicasting", RFC 1112, August 1989.

[6] Fenner, W., "Internet Group Management Protocol, Version 2", RFC 2236, November 1997.

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

[8] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, August 2006.

Author’s Address

Hitoshi Asaeda
National Institute of Information and Communications Technology
4-2-1 Nukui-Kitamachi
Koganei, Tokyo 184-8795
Japan

Email: asaeda@nict.go.jp

Network Working Group
Internet-Draft
Updates: 5384 (if approved)
Intended status: Standards Track
Expires: January 3, 2016

S. Venaas
J. Arango
Cisco Systems
I. Kouvelas

July 2, 2015

Hierarchical Join/Prune Attributes
draft-ietf-pim-hierarchicaljoinattr-04.txt

Abstract

This document defines a hierachical method of encoding Join attributes, providing a more efficient encoding when the same attribute values need to be specified for multiple sources in a PIM Join/Prune message.

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 January 3, 2016.

Copyright Notice

Table of Contents

1. Introduction

PIM Join attributes as defined in [RFC5384] allow for specifying a set of attributes for each of the joined or pruned sources in a PIM Join/Prune message. Attributes must be separately specified for each individual source in the message. However, in some cases the same attributes and values need to be specified for some, or even all, the sources in the message. The attributes and their values then need to be repeated for each of the sources where they apply.

This document provides a hierarchical way of encoding attributes and their values in a Join/Prune message, so that if the same attribute and value is to apply for all the sources, it needs only be specified once in the message. Similarly, if all the sources in a specific group set share a specific attribute and value, it needs only be specified once for the entire group set.

This document updates [RFC5384] which defines an encoding to be used for Encoded-Source Addresses. This document extends this by specifying the same encoding type also for Encoded-Unicast and Encoded-Group formats. This document defines a new IANA registry for PIM encoding types which is to be used for all the fields in PIM messages where encoding types are used, replacing the old registry that is specific to Encoded-Source Addresses. The encoding type used for Join attributes is however still limited to be used in Join/Prune messages. Note that Join attributes, as they are referred to in [RFC5384], also apply to pruned sources in a Join/Prune message. Thus the more correct name Join/Prune attributes will be used throughout the rest of this document.

This document allows Join/Prune attributes to be specified in the Upstream Neighbor Address field, and also in the Multicast Group Address field, of a Join/Prune message. It defines how this is used to specify the same Join/Prune attribute and value for multiple

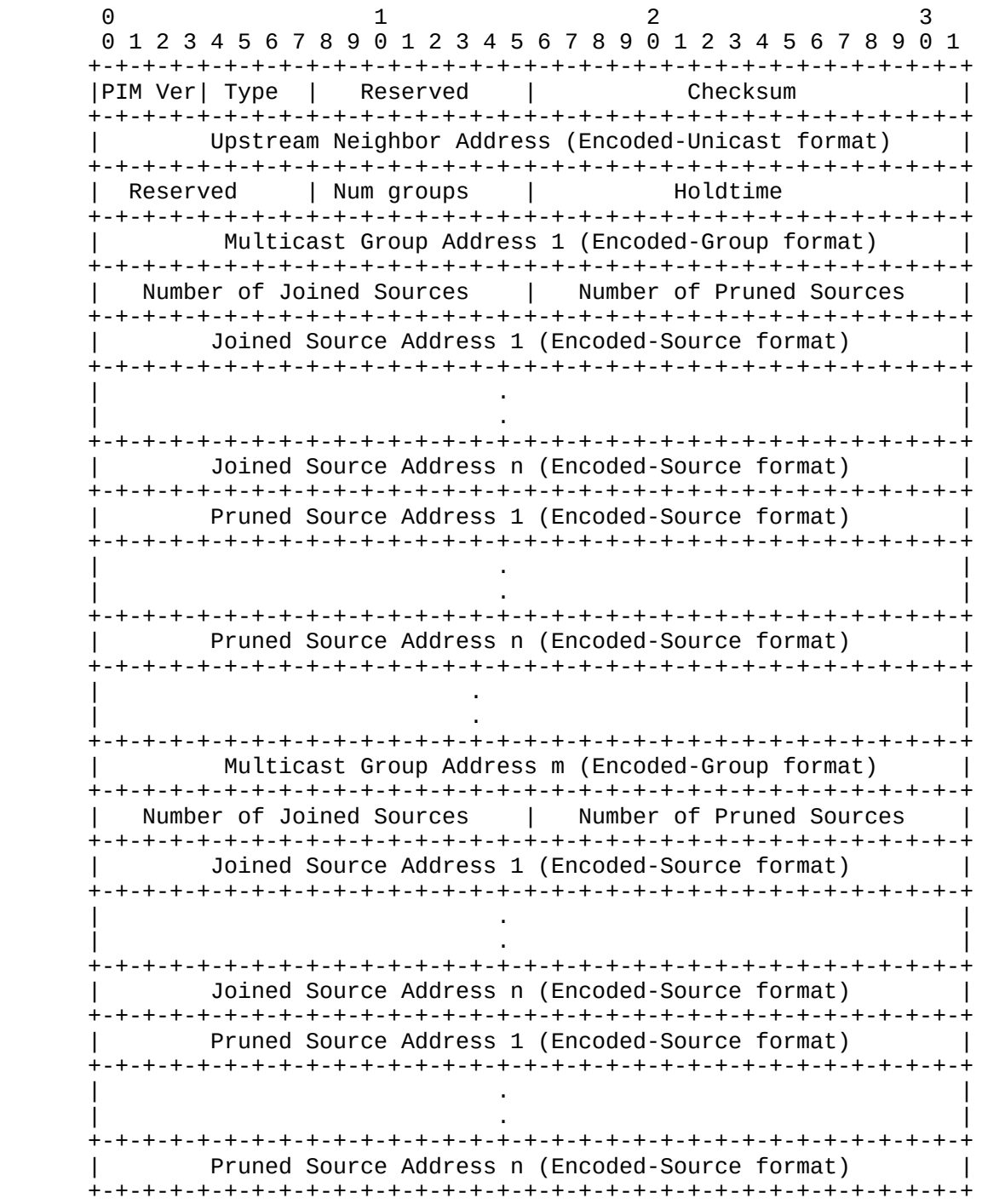
sources. This document also defines a new Hello Option to indicate support for the hierarchical encoding specified.

2. Requirements Notation

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

3. Hierarchical Join/Prune Attribute Definition

The format of a PIM Join/Prune message is defined in [RFC4601] as follows:


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
PIM Ver
Type
Reserved
Checksum
Upstream Neighbor Address (Encoded-Unicast format)
Reserved
Num groups
Holdtime
Multicast Group Address 1 (Encoded-Group format)
Number of Joined Sources
Number of Pruned Sources
Joined Source Address 1 (Encoded-Source format)
Joined Source Address n (Encoded-Source format)
Pruned Source Address 1 (Encoded-Source format)
Pruned Source Address n (Encoded-Source format)
Multicast Group Address m (Encoded-Group format)
Number of Joined Sources
Number of Pruned Sources
Joined Source Address 1 (Encoded-Source format)
Joined Source Address n (Encoded-Source format)
Pruned Source Address 1 (Encoded-Source format)
Pruned Source Address n (Encoded-Source format)

The message contains a single Upstream Neighbor Address, and one or more group sets. Each group set contains a Group Address and two source lists, the Joined Sources and the Pruned Sources. The Upstream Neighbor Address, the group addresses and the source addresses are encoded in Encoded-Unicast format, Encoded-Group format and Encoded-Source format, respectively. In this document we make use of this to allow Join/Prune attributes in each of these addresses, using the encoding in Section 4.

For a Join/Prune message we define a hierarchy of Join/Prune attributes. At the highest level, that is the least specific, we have attributes that apply to every source in the message. These are encoded in the Upstream Neighbor Address. At the next more specific level we have attributes that apply to every source in a group set. They are encoded in a Group Address. And finally at the most specific level, we have attributes that just apply to a single source, encoded in the source address as defined in [RFC5384].

The complete set of attributes that apply to a given source is obtained by combining the message wide attributes, the attributes of the group set that the source belongs to, and the source specific attributes. However, if the same attribute is specified at multiple levels, then the one at the most specific level overrides the other instances of the attribute.

Note that Join/Prune attributes are still applied to sources as specified in [RFC5384]. This document does not change the meaning of any attributes, it is simply a more compact way of encoding an attribute when the same attribute and value applies to multiple sources.

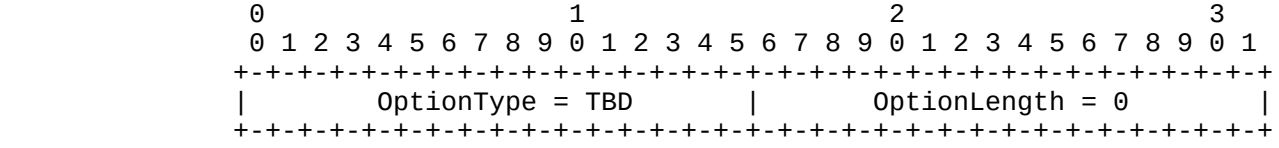
4. PIM Address Encoding Types

Addresses in PIM messages are specified together with an address family and an encoding type. This applies to Encoded-Unicast, Encoded-Group and Encoded-Source addresses. The encoding types allow the address to be encoded according to different schemes. While it is possible to have the same encoding type value indicate different encodings depending on whether it is a Unicast, Group or Source address, it is simpler to have the same encoding type value indicate the same encoding independent of where it is used. This means that as currently defined, 0 is native encoding, and 1 is Join/Prune attributes encoding, encoded according to [RFC5384]. Even if the encoding type space is shared between the different address types (Encoded-Unicast, Encoded-Group and Encoded-Source), one could have a specific encoding apply to a specific address type if needed.

The current IANA PIM Encoded-Source Address Encoding Type Field registry should be changed into a PIM Address Encoding Type registry.

5. Hierarchical Join/Prune Attribute Hello Option

A PIM router indicates that it supports the mechanism specified in this document by including the Hierarchical Join/Prune Attribute Hello Option in its PIM Hello message. Note that it also needs to include the Join-Attribute Hello option as specified in [RFC5384]. The format of the Hierarchical Join/Prune Attribute Hello Option is defined to be:



OptionType = TBD, OptionLength = 0. Note that there is no option value included.

A PIM router MUST NOT send a Join/Prune message with Join/Prune attributes encoded in the Upstream Neighbor Address or any of the group addresses out any interface on which there is a PIM neighbor that has not included this option in its Hellos. Even a router that is not the upstream neighbor must be able to parse the message in order to perform Join suppression and Prune override.

6. Security Considerations

This document specifies a more compact encoding of Join/Prune attributes. Use of the encoding has no impact on security.

7. IANA Considerations

The current PIM Encoded-Source Address Encoding Type Field registry should be changed into a PIM Address Encoding Type registry. The only required change is the name of the registry. The contents remain the same.

A new PIM Hello Option type needs to be assigned. The string TBD needs to be replaced with the permanently assigned value.

8. Acknowledgments

The authors would like to thank Siva Kollipara for providing feedback on the document.

9. Normative References

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

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

[RFC5384] Boers, A., Wijnands, I., and E. Rosen, "The Protocol Independent Multicast (PIM) Join Attribute Format", RFC 5384, November 2008.

Authors' Addresses

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

Email: stig@cisco.com

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

Email: jearango@cisco.com

Isidor Kouvelas

Email: isidor@kouvelas.net

PIM Working Group
Internet-Draft
Intended status: Experimental
Expires: April 21, 2016

LM. Contreras
Telefonica
CJ. Bernardos
Universidad Carlos III de Madrid
October 19, 2015

Requirements for the extension of the MLD proxy functionality to support multiple upstream interfaces
draft-ietf-pim-multiple-upstreams-reqs-01

Abstract

The purpose of this document is to define the requirements for a MLD (for IPv6) or IGMP (for IPv4) proxy with multiple interfaces covering a variety of applicability scenarios.

Table of Contents

1. Introduction

The aim of this document is to define the functionality that an IGMP/MLD proxy with multiple upstream interfaces should have in order to support different scenarios of applicability in both fixed and mobile networks. This compatibility is needed in order to simplify node functionality and to ensure an easier deployment of multicast capabilities in all the use cases described in this document.

2. Terminology

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

This document uses the terminology defined in RFC4605 [RFC4605]. Specifically, the definition of Upstream and Downstream interfaces, which are reproduced here for completeness.

Upstream interface: A proxy device's interface in the direction of the root of the tree. Also called the "Host interface".

Downstream interface: Each of a proxy device's interfaces that is not in the direction of the root of the tree. Also called the "Router interfaces".

3. Problem statement

The concept of IGMP/MLD proxy with several upstream interfaces has emerged as a way of optimizing (and in some cases enabling) service delivery scenarios where separate multicast service providers are reachable through the same access network infrastructure. Figure 1 presents the conceptual model under consideration.

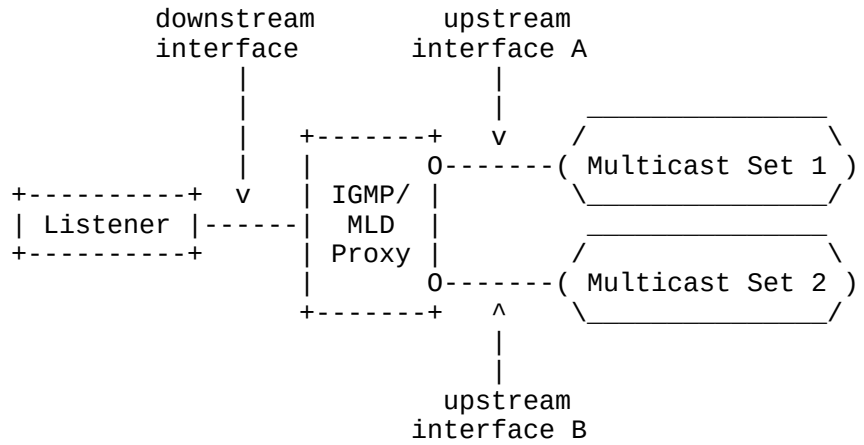


Figure 1: Concept of IGMP/MLD proxy with multiple upstream interfaces

The current version of this document is focused on fixed network scenarios. Applicability of IGMP/MLD proxies with multiple upstream interfaces in mobile environments has been previously described in RFC7287 [RFC7287]. Mobile network scenarios will be covered in future versions of this document.

In the case of fixed networks, multicast wholesale services in a competitive residential market require an efficient distribution of multicast traffic from different operators or content providers, i.e. the incumbent operator and a number of alternative providers, on the network infrastructure of the former. Existing proposals are based on the use of PIM routing from the metro/core network, and multicast traffic aggregation on the same tree. A different approach could be achieved with the use of an IGMP/MLD proxy with multiple upstream interfaces, each of them pointing to a distinct multicast router in the metro/core border which is part of separated multicast trees deep in the network. Figure 2 graphically describes this scenario.

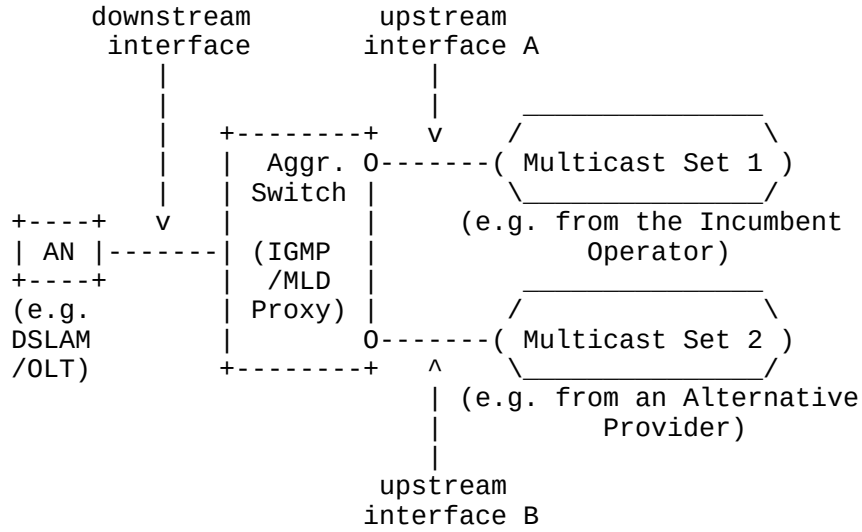


Figure 2: Example of usage of an IGMP/MLD proxy with multiple upstream interfaces in a fixed network scenario

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

4. Scenarios of applicability

Having multiple upstream interfaces creates a new decision space for delivering the proper multicast content to the subscriber. Basically it is now possible to implement channel-based or subscriber-based upstream selection, according to mechanisms or policies that could be defined for the multicast service provision.

This section describes in detail a number of scenarios of applicability of an IGMP/MLD proxy with multiple upstream interfaces in place. A number of requirements for the IGMP/MLD proxy functionality are identified from those scenarios.

4.1. Fixed network scenarios

Residential broadband users get access to multiple IP services through fixed network infrastructures. End user's equipment is connected to an access node, and the traffic of a number of access nodes is collected in aggregation switches.

For the multicast service, the use of an IGMP/MLD proxy with multiple upstream interfaces in those switches can provide service flexibility in a lightweight and simpler manner if compared with PIM-routing based alternatives.

4.1.1. Multicast wholesale offer for residential services

This scenario has been already introduced in the previous section, and can be seen in Figure 2. There are two different operators, the one operating the fixed network where the end user is connected (e.g., typically an incumbent operator), and the one providing the Internet service to the end user (e.g., an alternative Internet service provider). Both can offer multicast streams that can be subscribed by the end user, independently of which provider contributes with the content.

Note that it is assumed that both providers offer distinct multicast groups. However, more than one subscription to multicast channels of different providers could take place simultaneously.

4.1.1.1. Requirements

- The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding provider's multicast router.

- The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the providers to the corresponding end user.

4.1.2. Multicast resiliency

In current PIM-based solutions, the resiliency of the multicast distribution relays on the routing capabilities provided by protocols like PIM and VRRP RFC5798 [RFC5798]. A simpler scheme could be achieved by implementing different upstream interfaces on IGMP/MLD proxies, providing path diversity through the connection to distinct leaves of a given multicast tree.

It is assumed that only one of the upstream interfaces is active in receiving the multicast content, while the other is up and in standby mode for fast switching.

4.1.2.1. Requirements

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

- The IGMP/MLD proxy should be able to deliver multicast control messages received in the active upstream to the end users, while ignoring the control messages of the standby upstream interface.

- The IGMP/MLD proxy should be able of rapidly switching from the active to the standby upstream interface in case of network failure, transparently to the end user.

4.1.3. Load balancing for multicast traffic in the metro segment

A single upstream interface in existing IGMP/MLD proxy functionality typically forces the distribution of all the channels on the same path in the last segment of the network. Multiple upstream interfaces could naturally split the demand, alleviating the bandwidth requirements in the metro segment.

4.1.3.1. Requirements

- The IGMP/MLD proxy should be able to deliver multicast control messages sent by the end user to the corresponding multicast router which provides the channel of interest.

- The IGMP/MLD proxy should be able to deliver multicast control messages sent by each of the multicast routers to the corresponding end user.

- The IGMP/MLD proxy should be able to decide which upstream interface is selected for any new channel request according to defined criteria (e.g., load balancing).

4.1.4. Summary of the requirements needed for fixed network scenarios

Following the analysis above, a number of different requirements can be identified by the IGMP/MLD proxy to support multiple upstream interfaces in fixed network scenarios. The following table summarizes these requirements.

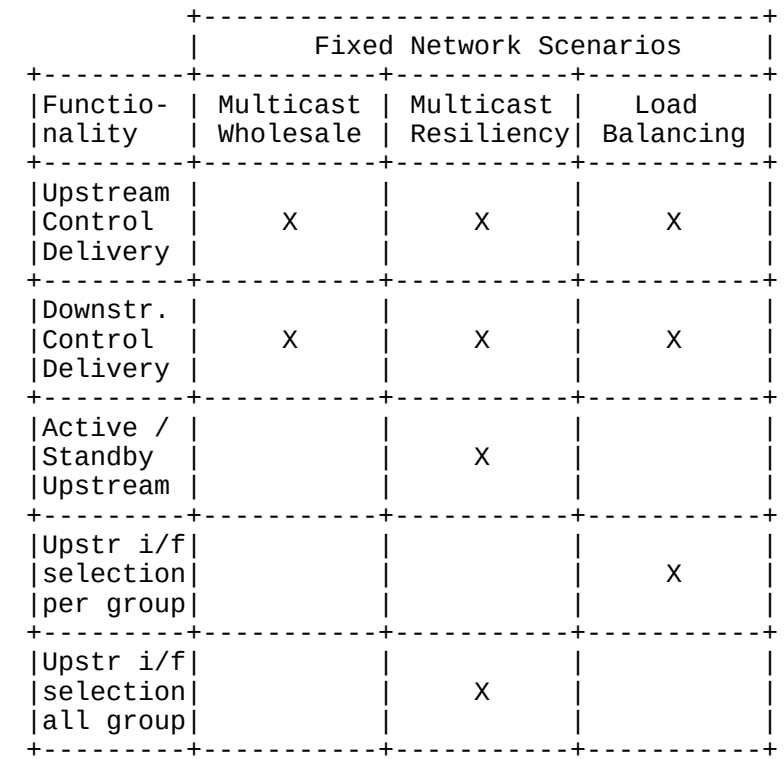


Figure 3: Functionality needed on IGMP/MLD proxy with multiple upstream interfaces per application scenario in fixed networks

4.2. Mobile network scenarios

To be done.

5. Security Considerations

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

6. IANA Considerations

To be completed

7. Acknowledgements

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

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>.

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <http://www.rfc-editor.org/info/rfc4605>.

8.2. Informative References

[RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <http://www.rfc-editor.org/info/rfc5798>.

[RFC7287] Schmidt, T., Ed., Gao, S., Zhang, H., and M. Waehlisch, "Mobile Multicast Sender Support in Proxy Mobile IPv6 (PMIPv6) Domains", RFC 7287, DOI 10.17487/RFC7287, June 2014, <http://www.rfc-editor.org/info/rfc7287>.

Authors' Addresses

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://people.tid.es/LuisM.Contreras/

Carlos J. Bernardos
Universidad Carlos III de Madrid
Av. Universidad, 30
Leganes, Madrid 28911
Spain

Phone: +34 91624 6236
Email: cjbc@it.uc3m.es
URI: http://www.it.uc3m.es/cjbc/

PIM Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

X. Liu
Ericsson
P. McAllister
Metaswitch Networks
A. Peter
Juniper Networks

October 19, 2015

A YANG data model for Protocol-Independent Multicast (PIM)
draft-mcallister-pim-yang-01

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 21, 2016.

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 [RFC4601], including the Source-Specfic subset [RFC3569], Dense Mode [RFC3973], and Bi-directional PIM [RFC5015].

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

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

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

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

2.2. Optional capabilities

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

specification is easy to express through some combination of the features in the base model and simple vendor augmentations.

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

The extensive use of feature declarations should also substantially simplify the capability negotiation process for a vendor's PIM implementation.

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

For the same reason, wide constant ranges (for example, timer maxima and minima) will be used in the model. It is expected that vendors will augment the model with any specific restrictions that might be required. Vendors may also extend the features list with proprietary extensions.

2.3. Top-level structure

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

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

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

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

2.4. Position of address family in hierarchy

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

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

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

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

3. Unresolved Issues

3.1. Current status of work in progress

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

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

3.2. Group range mappings

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

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

3.3. Issues blocked on other model designers

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

1. The position of BFD in the configuration; does the BFD model augment various different protocols, or do the protocols have to augment themselves individually to support BFD?

2. The abstract concepts of a "set of IP addresses" or "set of IP prefixes", as represented in implementations by prefixes, ACLs or policy statements, is a general concept out of the scope of this document. It will presumably have a yang data-type the instantiation of which is vendor-specific.

4. Module Structure

4.1. PIM base module

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

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

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

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

```
      |     +--rw duration?   uint16
      +--rw interfaces
         +--rw interface* [interface]
            +--rw interface         if:interface-ref
            +--rw address-family* [address-family]
               +--rw address-family       identityref
               +--rw dr-priority?         uint32 {intf-dr-priority}?
               +--rw hello-interval?      uint16 {intf-hello-interval}?
               +--rw (hello-holdtime-or-multipler)?
               |  +--:(holdtime) {intf-hello-holdtime}?
               |  |  +--rw hello-holdtime?      uint16
               |  +--:(multipler) {intf-hello-multipler}?
               |     +--rw hello-multipler?     uint8
               +--rw jp-interval?         uint16 {intf-jp-interval}?
               +--rw (jp-holdtime-or-multipler)?
               |  +--:(holdtime) {intf-jp-holdtime}?
               |  |  +--rw jp-holdtime?         uint16
               |  +--:(multipler) {intf-jp-multipler}?
               |     +--rw jp-multipler?        uint8
               +--rw propagation-delay?   uint16 {intf-propagation-delay}?
               +--rw override-interval?   uint16 {intf-override-interval}?
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols:
   +--ro pim
      +--ro address-family* [address-family]
      |  +--ro address-family        identityref
      |  +--ro statistics
      |  |  +--ro discontinuity-time?   yang:date-and-time
      |  |  +--ro error
      |  |  |  +--ro assert?                        yang:counter32
      |  |  |  +--ro bsr?                           yang:counter32
      |  |  |  +--ro candidate-rp-advertisement?    yang:counter32
      |  |  |  +--ro hello?                         yang:counter32
      |  |  |  +--ro join-prune?                    yang:counter32
      |  |  |  +--ro register?                      yang:counter32
      |  |  |  +--ro register-stop?                 yang:counter32
      |  |  |  +--ro state-refresh?                 yang:counter32
      |  |  +--ro queue
      |  |  |  +--ro size?       uint32
      |  |  |  +--ro overflow?   yang:counter32
      |  |  +--ro received
      |  |  |  +--ro assert?                        yang:counter32
      |  |  |  +--ro bsr?                           yang:counter32
      |  |  |  +--ro candidate-rp-advertisement?    yang:counter32
      |  |  |  +--ro hello?                         yang:counter32
      |  |  |  +--ro join-prune?                    yang:counter32
      |  |  |  +--ro register?                      yang:counter32
      |  |  |  +--ro register-stop?                 yang:counter32
      |  |  |  +--ro state-refresh?                 yang:counter32
```

```
|  |  +--ro sent
|  |     +--ro assert?                       yang:counter32
|  |     +--ro bsr?                          yang:counter32
|  |     +--ro candidate-rp-advertisement?   yang:counter32
|  |     +--ro hello?                        yang:counter32
|  |     +--ro join-prune?                   yang:counter32
|  |     +--ro register?                     yang:counter32
|  |     +--ro register-stop?                yang:counter32
|  |     +--ro state-refresh?                yang:counter32
|  +--ro topology-tree-info
|     +--ro ipv4-route* [group source-addr is-rpt]
|     |  +--ro group                 inet:ipv4-address
|     |  +--ro source-addr           union
|     |  +--ro is-rpt                boolean
|     |  +--ro expire?               uint32
|     |  +--ro incoming-interface?   if:interface-ref
|     |  +--ro mode?                 pim-mode
|     |  +--ro msdp-learned?         boolean
|     |  +--ro rp-address?           inet:ip-address
|     |  +--ro rpf-neighbor?         inet:ip-address
|     |  +--ro spt-bit?              boolean
|     |  +--ro up-time?              uint32
|     |  +--ro outgoing-interface* [name]
|     |     +--ro name        if:interface-ref
|     |     +--ro expire?     yang:timeticks
|     |     +--ro up-time?    yang:timeticks
|     |     +--ro jp-state?   enumeration
|     +--ro ipv6-route* [group source-addr is-rpt]
|        +--ro group                 inet:ipv6-address
|        +--ro source-addr           union
|        +--ro is-rpt                boolean
|        +--ro expire?               uint32
|        +--ro incoming-interface?   if:interface-ref
|        +--ro mode?                 pim-mode
|        +--ro msdp-learned?         boolean
|        +--ro rp-address?           inet:ip-address
|        +--ro rpf-neighbor?         inet:ip-address
|        +--ro spt-bit?              boolean
|        +--ro up-time?              uint32
|        +--ro outgoing-interface* [name]
|           +--ro name        if:interface-ref
|           +--ro expire?     yang:timeticks
|           +--ro up-time?    yang:timeticks
|           +--ro jp-state?   enumeration
+--ro interfaces
   +--ro interface* [interface]
      +--ro interface         if:interface-ref
      +--ro address-family* [address-family]
```

```
               +--ro address-family       identityref
               +--ro dr-priority?         uint32 {intf-dr-priority}?
               +--ro hello-interval?      uint16 {intf-hello-interval}?
               +--ro (hello-holdtime-or-multipler)?
               |  +--:(holdtime) {intf-hello-holdtime}?
               |  |  +--ro hello-holdtime?      uint16
               |  +--:(multipler) {intf-hello-multipler}?
               |     +--ro hello-multipler?     uint8
               +--ro jp-interval?         uint16 {intf-jp-interval}?
               +--ro (jp-holdtime-or-multipler)?
               |  +--:(holdtime) {intf-jp-holdtime}?
               |  |  +--ro jp-holdtime?         uint16
               |  +--:(multipler) {intf-jp-multipler}?
               |     +--ro jp-multipler?        uint8
               +--ro propagation-delay?   uint16 {intf-propagation-delay}?
               +--ro override-interval?   uint16 {intf-override-interval}?
               +--ro ipv4
               |  +--ro address*   inet:ipv4-address
               |  +--ro dr-addr?   inet:ipv4-address
               +--ro ipv6
               |  +--ro address*   inet:ipv6-address
               |  +--ro dr-addr?   inet:ipv6-address
               +--ro oper-status?         enumeration
               +--ro hello-expire?        uint32
               +--ro neighbor-ipv4* [address]
               |  +--ro address        inet:ipv4-address
               |  +--ro bfd-status?    enumeration
               |  +--ro expire?        uint32
               |  +--ro dr-priority?   uint32
               |  +--ro gen-id?        uint32
               |  +--ro up-time?       uint32
               +--ro neighbor-ipv6* [address]
                  +--ro address        inet:ipv6-address
                  +--ro bfd-status?    enumeration
                  +--ro expire?        uint32
                  +--ro dr-priority?   uint32
                  +--ro gen-id?        uint32
                  +--ro up-time?       uint32
notifications:
   +---n pim-neighbor-event
   |  +--ro event-type?                    neighbor-event-type
   |  +--ro routing-instance-state-ref?   rt:routing-instance-state-ref
   |  +--ro interface-state-ref?          leafref
   |  +--ro interface-af-state-ref?       leafref
   |  +--ro neighbor-ipv4-state-ref?      leafref
   |  +--ro neighbor-ipv6-state-ref?      leafref
   |  +--ro up-time?                      uint32
   +---n pim-interface-event
```

```
      +--ro event-type?                   interface-event-type
      +--ro routing-instance-state-ref?   rt:routing-instance-state-ref
      +--ro interface-state-ref?          leafref
      +--ro ipv4
      |  +--ro address*   inet:ipv4-address
      |  +--ro dr-addr?   inet:ipv4-address
      +--ro ipv6
         +--ro address*   inet:ipv6-address
         +--ro dr-addr?   inet:ipv6-address
```

4.2. PIM RP module

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

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

```
         +--rw rp-candidate-ipv6-address* [ipv6-address] {candidate-ipv6}?
            +--rw ipv6-address    inet:ipv6-address
            +--rw policy?         string
            +--rw mode?           identityref
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:
   +--ro rp
      +--ro bsr {bsr}?
      |  +--ro bsr-candidate!
      |  |  +--ro (interface-or-address)?
      |  |  |  +--:(interface) {candidate-interface}?
      |  |  |  |  +--ro interface           if:interface-ref
      |  |  |  +--:(ipv4-address) {candidate-ipv4}?
      |  |  |  |  +--ro ipv4-address        inet:ipv4-address
      |  |  |  +--:(ipv6-address) {candidate-ipv6}?
      |  |  |     +--ro ipv6-address        inet:ipv6-address
      |  |  +--ro hash-mask-length    uint8
      |  |  +--ro priority            uint8
      |  +--ro rp-candidate-interface* [interface] {candidate-interface}?
      |  |  +--ro interface    if:interface-ref
      |  |  +--ro policy?      string
      |  |  +--ro mode?        identityref
      |  +--ro rp-candidate-ipv4-address* [ipv4-address] {candidate-ipv4}?
      |  |  +--ro ipv4-address    inet:ipv4-address
      |  |  +--ro policy?         string
      |  |  +--ro mode?           identityref
      |  +--ro rp-candidate-ipv6-address* [ipv6-address] {candidate-ipv6}?
      |  |  +--ro ipv6-address    inet:ipv6-address
      |  |  +--ro policy?         string
      |  |  +--ro mode?           identityref
      |  +--ro bsr
      |  |  +--ro addr?               inet:ip-address
      |  |  +--ro hash-mask-length?   uint8
      |  |  +--ro priority?           uint8
      |  |  +--ro up-time?            uint32
      |  +--ro (election-state)? {bsr-election-state}?
      |  |  +--:(candidate)
      |  |  |  +--ro candidate-bsr-state?              enumeration
      |  |  +--:(non-candidate)
      |  |     +--ro non-candidate-bsr-state?          enumeration
      |  +--ro bsr-next-bootstrap?                uint16
      |  +--ro rp
      |  |  +--ro rp-address?     inet:ip-address
      |  |  +--ro group-policy?   string
      |  |  +--ro up-time?        uint32
      |  +--ro rp-candidate-next-advertisement?   uint16
      +--ro rp-list
      |  +--ro ipv4-rp* [ipv4-addr]
```

```
      |  |  +--ro ipv4-addr           inet:ipv4-address
      |  |  +--ro info-source-addr?   inet:ipv4-address
      |  |  +--ro info-source-type?   enumeration
      |  |  +--ro up-time?            uint32
      |  |  +--ro expire?             uint16
      |  +--ro ipv6-rp* [ipv6-addr]
      |     +--ro ipv6-addr           inet:ipv6-address
      |     +--ro info-source-addr?   inet:ipv6-address
      |     +--ro info-source-type?   enumeration
      |     +--ro up-time?            uint32
      |     +--ro expire?             uint16
      +--ro rp-mappings
         +--ro ipv4-rp* [group rp-addr]
         |  +--ro group                inet:ipv4-prefix
         |  +--ro rp-addr              inet:ipv4-address
         |  +--ro info-source-addr?    inet:ipv4-address
         |  +--ro info-source-type?    enumeration
         |  +--ro up-time?             uint32
         |  +--ro expire?              uint16
         +--ro ipv6-rp* [group rp-addr]
            +--ro group                inet:ipv6-prefix
            +--ro rp-addr              inet:ipv6-address
            +--ro info-source-addr?    inet:ipv6-address
            +--ro info-source-type?    enumeration
            +--ro up-time?             uint32
            +--ro expire?              uint16
notifications:
   +---n pim-rp-event
      +--ro event-type?                    rp-event-type
      +--ro routing-instance-state-ref?    rt:routing-instance-state-ref
      +--ro instance-af-state-ref?         leafref
      +--ro group?                         inet:ip-address
      +--ro rp-address?                    inet:ip-address
      +--ro is-rpt?                        boolean
      +--ro mode?                          pim-base:pim-mode
      +--ro message-origin?                inet:ip-address
```

4.3. PIM-SM module

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

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

4.4. PIM-DM module

This module will cover Dense Mode configuration.

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

4.5. PIM-BIDIR module

This module will cover Bidirectional PIM configuration.

```
module: ietf-pim-bidir
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:
   +--rw bidir
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:interfaces/pim-base:interface/
pim-base:address-family:
   +--rw bidir!
      +--rw df-election {intf-df-election}?
         +--rw offer-interval?     uint32
         +--rw backoff-interval?   uint32
         +--rw offer-multipler?    uint8
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp/pim-rp:static-rp/
pim-rp:ipv4-rp:
   +--rw bidir!
      +--rw policy-name?   string
      +--rw override?      boolean {static-rp-override}?
augment /rt:routing/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family/pim-rp:rp/pim-rp:static-rp/
pim-rp:ipv6-rp:
   +--rw bidir!
      +--rw policy-name?   string
      +--rw override?      boolean {static-rp-override}?
augment /rt:routing-state/rt:routing-instance/rt:routing-protocols/
pim-base:pim/pim-base:address-family:
   +--ro bidir
```

5. PIM YANG Modules

5.1. PIM base module

```
<CODE BEGINS>

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

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

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

  import ietf-interfaces {
    prefix "if";
  }

  import ietf-routing {
    prefix "rt";
  }

  organization
    "IETF PIM Working Group";

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

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

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

     Editors:   ";

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

  revision 2015-10-08 {
```

```
      description
        "Initial revision.";
      reference
        "RFC XXXX: A YANG Data Model for PIM";
    }

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

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

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

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

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

    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-multipler {
      description
        "Support configuration of interface join prune multipler.";
```

```
    }

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

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

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

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

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

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

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

    typedef timer-value {
      type union {
        type uint16;
        type enumeration {
          enum "infinity" {
            description "The timer is set to infinity.";
          }
          enum "no-expiry" {
```

```
            description "The timer is not set.";
          }
        }
      }
      units seconds;
      description "Timer value type.";
    } // timer-value

    /*
     * Identities
     */

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

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

    grouping interface-config-attributes {
      description
        "A grouping defining interface attributes.";
      leaf dr-priority {
```

```
          if-feature intf-dr-priority;
          type uint32;
          description "DR priority";
        }
        leaf hello-interval {
          if-feature intf-hello-interval;
          type uint16;
          description "Hello interval";
        }
        choice hello-holdtime-or-multipler {
          description "Use holdtime or multipler";
          case holdtime {
            if-feature intf-hello-holdtime;
            leaf hello-holdtime {
              type uint16;
              description "Hello holdtime";
            }
          }
          case multipler {
            if-feature intf-hello-multipler;
            leaf hello-multipler {
              type uint8;
              description "Hello multipler";
            }
          }
        }
        leaf jp-interval {
          if-feature intf-jp-interval;
          type uint16;
          description "Join prune interval";
        }
        choice jp-holdtime-or-multipler {
          description "Use holdtime or multipler";
          case holdtime {
            if-feature intf-jp-holdtime;
            leaf jp-holdtime {
              type uint16;
              description "Join prune holdtime";
            }
          }
          case multipler {
            if-feature intf-jp-multipler;
            leaf jp-multipler {
              type uint8;
              description "Join prune multipler";
            }
          }
        }
```

```
        leaf propagation-delay {
          if-feature intf-propagation-delay;
          type uint16;
          description "Propagation description";
        }
        leaf override-interval {
          if-feature intf-override-interval;
          type uint16;
          description "Override interval";
        }
      } // interface-config-attributes

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

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

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

       leaf hello-expire {
         type uint32;
         description "Hello interval exiration time.";
       }

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

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

```
        uses neighbor-state-af-attributes;
      } // list neighbor-ipv4
    } // interface-state-af-attributes

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

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

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

        leaf expire {
          type yang:timeticks;
          description "Expiring information.";
        }

        leaf up-time {
          type yang:timeticks;
          description "";
        }

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

    grouping neighbor-state-af-attributes {
      description
        "A grouping defining neighbor per af attributes.";
      leaf bfd-status {
        type enumeration {
          enum up {
            description
            "";
```

```
            }
            enum down {
              description
              "";
            }
          }
          description "";
        }
        leaf expire {
          type uint32;
          units seconds;
          description "Neighbor expiring information.";
        }
        leaf dr-priority {
          type uint32;
          description "DR priority";
        }
        leaf gen-id {
          type uint32;
          description "Generation ID.";
        }
        leaf up-time {
          type uint32;
          units seconds;
          description "";
        }
      } // neighbor-state-af-attributes

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

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

      grouping pim-instance-af-state-ref {
        description
```

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

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

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

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

    grouping statistics-container {
      description
        "A container defining statistics attributes.";
      container statistics {
        description "";
        leaf discontinuity-time {
          type yang:date-and-time;
          description
            "The time on the most recent occasion at which any one
```

```
            or more of the statistic counters suffered a
            discontinuity. If no such discontinuities have occurred
            since the last re-initialization of the local
            management subsystem, then this node contains the time
            the local management subsystem re-initialized itself.";
        }
        container error {
          description "";
          uses statistics-error;
        }
        container queue {
          description "";
          uses statistics-queue;
        }
        container received {
          description "";
          uses statistics-sent-received;
        }
        container sent {
          description "";
          uses statistics-sent-received;
        }
      }
    } // statistics-container

    grouping statistics-error {
      description
        "A grouping defining error statistics
        attributes.";
      uses statistics-sent-received;
    } // statistics-error

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

```
     grouping statistics-sent-received {
       description
         "A grouping defining sent and received statistics
         attributes.";
       leaf assert {
         type yang:counter32;
         description
           "The number of assert messages.";
       }
       leaf bsr {
         type yang:counter32;
         description
           "The number of bsr messages.";
       }
       leaf candidate-rp-advertisement {
         type yang:counter32;
         description
           "The number of Candidate-RP-advertisement messages.";
       }
       leaf hello {
         type yang:counter32;
         description
           "The number of hello messages.";
       }
       leaf join-prune {
         type yang:counter32;
         description
           "The number of join/prune messages.";
       }
       leaf register {
         type yang:counter32;
         description
           "The number of register messages.";
       }
       leaf register-stop {
         type yang:counter32;
         description
           "The number of register stop messages.";
       }
       leaf state-refresh {
         type yang:counter32;
         description
           "The number of state refresh messages.";
       }
     } // statistics-sent-received

     /*
      * Configuration data nodes
```

```
     */

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

      container pim {
        description
          "PIM configuration data.";

        uses global-attributes;

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

        } // address-family

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

    /*
```

```
      * Operational state data nodes
      */

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

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

           uses statistics-container;

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

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

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

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

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

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

    /*
     * RPCs
     */

    /*
     * Notifications
     */
    notification pim-neighbor-event {
      description "Notification event for neighbor.";
      leaf event-type {
        type neighbor-event-type;
        description "Event type.";
      }
      uses pim-neighbor-state-ref;
      leaf up-time {
        type uint32;
        units seconds;
        description "";
      }
    }
    notification pim-interface-event {
      description "Notification event for interface.";
      leaf event-type {
        type interface-event-type;
        description "Event type.";
      }
      uses pim-interface-state-ref;
      container ipv4 {
        description "";
        leaf-list address {
          type inet:ipv4-address;
          description "";
        }
        leaf dr-addr {
          type inet:ipv4-address;
```

```
          description "";
        }
      }
      container ipv6 {
        description "";
        leaf-list address {
          type inet:ipv6-address;
          description "";
        }
        leaf dr-addr {
          type inet:ipv6-address;
          description "";
        }
      }
    }
  }
  <CODE ENDS>
```

5.2. PIM RP module

```
  <CODE BEGINS>

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

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

    import ietf-interfaces {
      prefix "if";
    }

    import ietf-routing {
      prefix "rt";
    }

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

    organization
      "IETF PIM Working Group";
```

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

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

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

        Editors:   ";

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

     revision 2015-10-08 {
       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).";
     }

     /*
```

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

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

```
          }
        }

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

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

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

      list rp-candidate-ipv4-address {
        when "../../../address-family = 'rt:ipv4'" {
          description
            "Only applicable to ipv4 address family.";
        }
        if-feature candidate-ipv4;
        key "ipv4-address";
        description
          "A list of RP candidates";
        leaf ipv4-address {
```

```
          type inet:ipv4-address;
          description
            "IPv4 address that the RP candidate uses.";
        }
        uses rp-candidate-attributes;
      }

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

    grouping bsr-state-attributes {
      description
        "Gouring of BSR state attributes.";
      container bsr {
        description
          "BSR information.";
        leaf addr {
          type inet:ip-address;
          description "BSR address";
        }
        leaf hash-mask-length {
          type uint8;
          description "";
        }
        leaf priority {
          type uint8 {
            range "0..255";
          }
          description "";
        }
        leaf up-time {
          type uint32;
          units seconds;
```

```
          description "";
        }
      }
      choice election-state {
        if-feature bsr-election-state;
        description "BSR election state.";
        case candidate {
          leaf candidate-bsr-state {
            type enumeration {
              enum "candidate" {
                description
                  "The router is a candidate to be the BSR for the
                   scope zone, but currently another router is the
                   preferred BSR.";
              }
              enum "pending" {
                description
                  "The router is a candidate to be the BSR for the
                   scope zone. Currently, no other router is the
                   preferred BSR, but this router is not yet the
                   elected BSR.  This is a temporary state that
                   prevents rapid thrashing of the choice of BSR
                   during BSR election.";
              }
              enum "elected" {
                description
                  "The router is the elected BSR for the scope zone
                   and it must perform all the BSR functions.";
              }
            }
            description
              "Candidate-BSR state.";
            reference
              "RFC5059, Section 3.1.1.";
          }
        }
        case "non-candidate" {
          leaf non-candidate-bsr-state {
            type enumeration {
              enum "no-info" {
                description
                  "The router has no information about this scope
                   zone.";
              }
              enum "accept-any" {
                description
                  "The router does not know of an active BSR, and will
                   accept the first Bootstrap message it sees as giving
```

```
                    the new BSR's identity and the RP-Set.";
                }
                enum "accept" {
                  description
                    "The router knows the identity of the current BSR,
                     and is using the RP-Set provided by that BSR. Only
                     Bootstrap messages from that BSR or from a C-BSR
                     with higher weight than the current BSR will be
                     accepted.";
                }
              }
              description
                "Non-candidate-BSR state.";
              reference
                "RFC5059, Section 3.1.2.";
            }
          }
        } // election-state
        leaf bsr-next-bootstrap {
          type uint16;
          units seconds;
          description "";
        }

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

      grouping rp-state-attributes {
```

```
      description
        "Gouring of static RP attributes.";
      leaf info-source-type {
        type enumeration {
          enum static {
            description
            "";
          }
          enum bootstrap {
            description
            "";
          }
        }
        description "";
      } // info-source-type
      leaf up-time {
        type uint32;
        units seconds;
        description "";
      }
      leaf expire {
        type uint16;
        units seconds;
        description "";
      }
    } // rp-state-attributes

    grouping static-rp-attributes {
      description
        "Gouring 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 rp-candidate-attributes {
      description
        "Gouring of RP candidate attributes.";
```

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

      /*
       * Configuration data nodes
       */

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

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

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

            list ipv6-rp {
              when "../../../address-family = 'rt:ipv6'" {
                description
```

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

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

    /*
     * Operational state data nodes
     */

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

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

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

        container rp-list {
          description
            "Containing a list RPs.";
          list ipv4-rp {
```

```
            when "../../../address-family = 'rt:ipv4'" {
              description
                "Only applicable to ipv4 address family.";
            }
            key "ipv4-addr";
            description
              "A list of IPv4 RP addresses.";
            leaf ipv4-addr {
              type inet:ipv4-address;
              description
                "RP address.";
            }
            leaf info-source-addr {
              type inet:ipv4-address;
              description
                "The address where RP information is learned.";
            }
            uses rp-state-attributes;
          }

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

        container rp-mappings {
          description
            "Containing a list group-to-RP mappings.";
          list ipv4-rp {
            when "../../../address-family = 'rt:ipv4'" {
              description
```

```
                "Only applicable to ipv4 address family.";
            }
            key "group rp-addr";
            description
              "A list of group-to-RP mappings.";
            leaf group {
              type inet:ipv4-prefix;
              description
                "Group prefix.";
            }
            leaf rp-addr {
              type inet:ipv4-address;
              description
                "RP address.";
            }
            leaf info-source-addr {
              type inet:ipv4-address;
              description
                "The address where RP information is learned.";
            }
            uses rp-state-attributes;
          }

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

```
        } // rp-mappings
      } // rp
    } // augment

    /*
     * RPCs
     */

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

5.3. PIM-SM module

```
<CODE BEGINS>

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

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

  import ietf-routing {
    prefix "rt";
  }

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

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

  organization
    "IETF PIM Working Group";

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

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

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

     Editors:   ";

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

  revision 2015-10-08 {
    description
      "Initial revision.";
    reference
```

```
        "RFC XXXX: A YANG Data Model for PIM";
    }

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

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

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

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

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

    /*
     * Configuration data nodes
     */
```

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

       container sm {
         description
           "PIM SM configuration data.";

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

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

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

                 leaf rp-addr {
                   type inet:ipv4-address;
                   description
                     "IP address of the router configured with anycast
                     RP. This is the IP address where the Register
                     messages are forwarded.";
                 }
               }
             }
```

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

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

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

```
        } // asm

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

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

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

      container sm {
        presence "Present to enable sparse-mode.";
        description
          "PIM SM configuration data.";

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

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

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

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

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

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

    /*
     * Operational state data nodes
     */

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

      container sm {
        description
          "PIM SM state data.";
      } // sm
    } // augment

    /*
     * RPCs
     */

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

5.4. PIM-DM module

```
  <CODE BEGINS>

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

    import ietf-routing {
      prefix "rt";
```

```
     }

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

     organization
       "IETF PIM Working Group";

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

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

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

        Editors:   ";

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

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

     /*
      * Features
      */

     /*
      * Identities
      */

     /*
      * Groupings
      */

     /*
      * Configuration data nodes
      */

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

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

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

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

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

    /*
     * Operational state data nodes
     */

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

    /*
     * RPCs
     */

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

5.5. PIM-BIDIR module

```
<CODE BEGINS>

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

  import ietf-routing {
    prefix "rt";
  }

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

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

  organization
    "IETF PIM Working Group";

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

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

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

     Editors:   ";

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

  revision 2015-10-08 {
    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.";
     }

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

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

     /*
      * Configuration data nodes
      */

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

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

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

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

    container df-election {
      if-feature intf-df-election;
      description
        "DF election attributes.";
      leaf offer-interval {
        type uint32;
        description "Offer interval";
      }
      leaf backoff-interval {
        type uint32;
        description "Backoff interval";
      }
      leaf offer-multipler {
        type uint8;
        description "Offer multipler";
      }
    } // df-election
  } // bidir
} // augment

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

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

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

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

```
    /*
     * Operational state data nodes
     */

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

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

    /*
     * RPCs
     */

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

6. TODO list

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

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

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

7. Security Considerations

The data model defined does not introduce any security implications.

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

8. IANA Considerations

TBD

9. Acknowledgements

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

10. References

10.1. Normative References

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

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <http://www.rfc-editor.org/info/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[I-D.ietf-netmod-rfc6087bis] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", draft-ietf-netmod-rfc6087bis-04 (work in progress), July 2015.

10.2. Informative References

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

[RFC3973] Adams, A., Nicholas, J., and W. Siadak, "Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)", RFC 3973, DOI 10.17487/RFC3973, January 2005, <http://www.rfc-editor.org/info/rfc3973>.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <http://www.rfc-editor.org/info/rfc4601>.

[RFC4610] Farinacci, D. and Y. Cai, "Anycast-RP Using Protocol Independent Multicast (PIM)", RFC 4610, DOI 10.17487/RFC4610, August 2006, <http://www.rfc-editor.org/info/rfc4610>.

[RFC5015] Handley, M., Kouvelas, I., Speakman, T., and L. Vicisano, "Bidirectional Protocol Independent Multicast (BIDIR-PIM)", RFC 5015, DOI 10.17487/RFC5015, October 2007, <http://www.rfc-editor.org/info/rfc5015>.

[RFC5059] Bhaskar, N., Gall, A., Lingard, J., and S. Venaas, "Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)", RFC 5059, DOI 10.17487/RFC5059, January 2008, <http://www.rfc-editor.org/info/rfc5059>.

[RFC6087] Bierman, A., "Guidelines for Authors and Reviewers of YANG Data Model Documents", RFC 6087, DOI 10.17487/RFC6087, January 2011, <http://www.rfc-editor.org/info/rfc6087>.

[I-D.ietf-netmod-routing-cfg] Lhotka, L. and A. Lindem, "A YANG Data Model for Routing Management", draft-ietf-netmod-routing-cfg-20 (work in progress), October 2015.

Authors' Addresses

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

EMail: xufeng.liu@ericsson.com

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

EMail: pete.mcallister@metaswitch.com

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

EMail: anishp@juniper.net

BIER Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

IJ. Wijnands
P. Pfister
Cisco Systems

October 19, 2015

Generic Multicast Router Election on LAN's
draft-wijnands-bier-mld-lan-election-00.txt

Abstract

When a host is connected to multiple multicast capable routers, each of these routers is a candidate to process the multicast flow for that LAN, but only one router should be elected to process it. This document proposes a generic multicast router election mechanism using Internet Group Management Protocol (IGMP) and Multicast Listener Discovery (MLD) that can be used by any Multicast Overlay Signalling Protocol (MOSP). Having such generic election mechanism removes a dependency on Protocol Independent Multicast (PIM).

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 21, 2016.

Table of Contents

1. Introduction

Hosts connected to Local Area Networks (LAN) use Internet Group Management Protocol (IGMP) [RFC4605] or Multicast Listener Discovery (MLD) [RFC3810] to report their interest in a particular multicast flow. A multicast flow is identified by a Group or a combination of Group and Source address. Routers connected to a LAN listen to these membership reports and signal that information to the Multicast Overlay Signalling Protocol (MOSP). When a host is connected to multiple routers, each of these routers is a candidate to forward the multicast flow onto that LAN, but only one of them should forward the packets for a given flow to avoid duplication of Multicast packets. A similar requirement exists for hosts that are sending multicast traffic and are connected to multiple routers on a LAN. If multiple routers accept the multicast packets from the LAN, duplication may occur and/or routing loops may be created.

Protocol Independent Multicast (PIM) [RFC4601] is a MOSP and has a built-in mechanism to elect a Designated Router (DR) on the receiver LAN and a Designated Forwarder (DF) on the senders LAN. The DR/DF election avoids duplication and looping of multicast packets. Other existing or candidate MOSPs, like Border Gateway Protocol (BGP) [RFC6514], Multi-point Label Distribution Protocols (mLDP) [RFC6826], Locator ID Seperation Protocol (LISP) [RFC6830] and IGMP/MLD [I-D.pfister-bier-mld] have no embedded LAN DR/DF election mechanism. These MOSPs still rely on PIM to perform DR/DF election on LANs.

With the introduction of mLDP and Bit Indexed Explicit Replication (BIER) [I-D.ietf-bier-architecture], there is no dependency on PIM to

transport multicast packets through the network. Having a dependency on PIM just for DR/DF election is undesirable if PIM is not selected as the MOSP. This document proposes a generic DR/DF election which can be used by any MOSP without having a dependency on PIM. It potentially allows for different MOSPs to coexistence on single LANs.

2. Terminology and Definitions

Readers of this document are assumed to be familiar with the terminology and concepts of the documents listed as Normative References. For convenience, some of the more frequently used terms appear below.

LAN:
: Local Area Network.

IGMP:
: Internet Group Management Protocol.

MLD:
: Multicast Listener Discovery.

mLDP:
: Multipoint LDP.

PIM:
: Protocol Independent Multicast.

ASM:
: Any Source Multicast.

RP:
: The PIM Rendezvous Point.

LISP:
: Locator ID Seperation Protocol.

BIER:
: Bit Indexed Explicit Replication.

MOSP:
: Multicast Overlay Signalling Protocol. This is a protocol that is (potentially) capable of announcing multicast flow membership across the network between multicast routers. For example PIM, mLDP, BGP, IGMP, MLD and LISP.

3. Specification of Requirements

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].

4. Problem Statement

In the following sections we describe the requirements for DR/DF election in more detail for hosts that are multicast senders and receivers connected to multiple routers on a single LAN.

4.1. Receiver side

Consider the network below in Topology1.

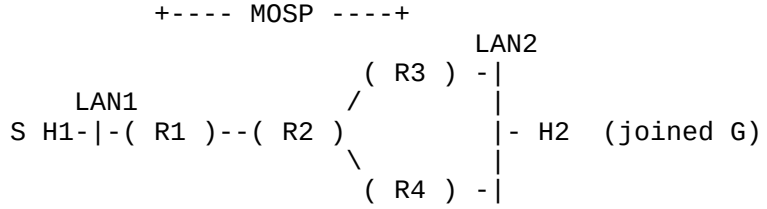


Figure 1

Suppose that H2 on LAN2 is joining a multicast Group G. The MOSP runs between R1, R3 and R4. Both R3 and R4 will receive the IGMP/MLD report, but only one of these should become the DR. One might consider that this problem can be detected and resolved by the MOSP. The MOSP could be enhanced to allow R1 to detect that both R2 and R4 are connected to the same LAN, and select only to forward the multicast flow to R3. That would solve the problem in the above topology, but would fail in the topology below:

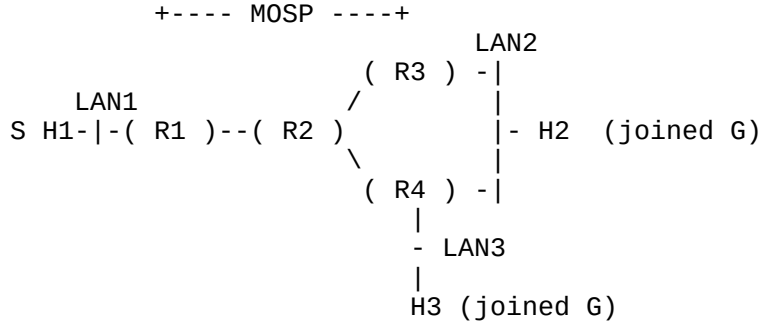


Figure 2

Consider that H3 on LAN3 joined the same multicast Group G. Since H3 is singly connected to R4, router R1 needs to forward the multicast flow to R4 in order for H3 to receive the packets. R4 does not have enough information to determine whether or not to forward on LAN2 for H2 when it receives the multicast packets due to H3. In other words, R4 needs DR state to avoid sending packets to H2 on LAN2.

4.2. Sender side

Consider the network below in Topology3.

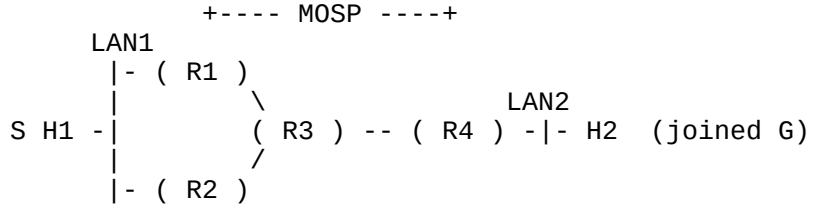


Figure 3

H1 is directly connected via a LAN1 to R1 and R2. H2 joins a multicast Group G, without specifying the Source. This is called Any Source Multicast (ASM). The MOSP signals R4s interest in Group G to R1 and R2. Note that there is no PIM deployed in this network and there is no Rendezvous Point (RP) that is a target for this receiving this Group membership. R4 has no information which routers in this network have multicast packets to sent for this Group. Since this is ASM, there may be multiple senders for this Group and H2 wants to receive them all. For that reason, R4 will use the MOSP to announce the membership to all edge nodes in the network (R1 and R2). This poses a potential problem since R1 and R2 are both directly connected to the Source S. If both R1 and R2 will forward the multicast packets to R4, H2 will receive duplicate packets. This is a problem that only occurs when a Source is dually connected to two or more routers connected to the BIER domain. This problem can be resolved by doing a Designated Forwarder (DF) election, similar to the DR election. If R1 and R2 are aware they are directly connected, an election will cause only one of them to forward the multicast packets into the BIER domain for a given (S,G) flow.

5. Proposal

As explain in Section 4, it is desirable to have a generic DR/DF election mechanism that can be used for existing and candidate MOSPs. Also, if a mix of MOSPs is used in the network, it is not obvious which MOSP is responsible for electing the DR/DF. If a single DR/DF is to be elected, and each MOSP does its own election, the MOSPs have

to negotiate among each other who will be responsible for DR/DF on a LAN. Independent of the MOSP, a single router connected to the LAN should be elected. It seems inefficient and unpractical to have each MOSP implement its own DR/DF election mechanism.

There is a process in the router that all the MOSPs depend on, that is the IGMP/MLD process. The DR/DF election is typically based on the Group address or Group and Source address of the multicast flow. This information is available in the IGMP/MLD process. In this document we propose to enhance the IGMP/MLD protocol to allow a DR/DF election among multicast routers connected to a LAN. As soon as a router is elected as DR/DF, it can select the MOSP that will be responsible to deliver the multicast flow to this router, and onwards onto the LAN(s).

IGMP/MLD has support for electing a Membership Querier based on the lowest IP address of the multicast routers sending out Membership Queries. It would be possible to use the elected Membership Querier as the DR/DF on a LAN. However, the authors believe that the Membership Querier procedures are not robust and extensible enough to be used DR/DF election on LANs. For example, if a new multicast router becomes active on a LAN, it will immediately assume the role of a Membership Querier, which can lead to duplication and/or looping of packets if also used as DR/DF. This duplication/looping will last until it learns about other Membership queriers with a lower IP address. Having two Membership queriers on the LAN has limited impact on the IGMP/MLD protocol it self, it would only cause more Membership Reports to be received.

The exact procedures to form a neighborship between IGMP/MLD routers will added in a later revision of this document.

6. Security Considerations

TBD.

7. IANA Considerations

TBD.

8. Acknowledgments

Many thanks to Neale Ranns and Greg Shepherd for their comments on this draft.

9. Normative References

[I-D.ietf-bier-architecture]
Wijnands, I., Rosen, E., Dolganow, A., Przygienda, T., and S. Aldrin, "Multicast using Bit Index Explicit Replication", draft-ietf-bier-architecture-02 (work in progress), July 2015.

[I-D.pfister-bier-mld]
Pfister, P., Wijnands, I., and M. Stenberg, "BIER Ingress Multicast Flow Overlay using Multicast Listener Discovery Protocols", draft-pfister-bier-mld-00 (work in progress), July 2015.

[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>.

[RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <http://www.rfc-editor.org/info/rfc3810>.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, DOI 10.17487/RFC4601, August 2006, <http://www.rfc-editor.org/info/rfc4601>.

[RFC4605] Fenner, B., He, H., Haberman, B., and H. Sandick, "Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying")", RFC 4605, DOI 10.17487/RFC4605, August 2006, <http://www.rfc-editor.org/info/rfc4605>.

[RFC6514] Aggarwal, R., Rosen, E., Morin, T., and Y. Rekhter, "BGP Encodings and Procedures for Multicast in MPLS/BGP IP VPNs", RFC 6514, DOI 10.17487/RFC6514, February 2012, <http://www.rfc-editor.org/info/rfc6514>.

[RFC6826] Wijnands, IJ., Ed., Eckert, T., Leymann, N., and M. Napierala, "Multipoint LDP In-Band Signaling for Point-to-Multipoint and Multipoint-to-Multipoint Label Switched Paths", RFC 6826, DOI 10.17487/RFC6826, January 2013, <http://www.rfc-editor.org/info/rfc6826>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <http://www.rfc-editor.org/info/rfc6830>.

Authors’ Addresses

IJsbrand Wijnands
Cisco Systems
De Kleetlaan 6a
Diegem 1831
Belgium

Email: ice@cisco.com

Pierre Pfister
Cisco Systems
Paris
France

Email: pierre.pfister@darou.fr

PIM WG
Internet-Draft
Intended status: Standards Track
Expires: January 20, 2016

Sandy. Zhang
Fangwei. Hu
BenChong. Xu
ZTE Corporation

July 19, 2015

PIM DR IMPROVEMENT
draft-zhang-pim-dr-improvement-00.txt

Abstract

PIM is worldly deployed multicast protocol. This document will improve the stability of PIM protocol, decrease the lost of multicast packets when the PIM DR (Designed Router) is down.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 20, 2016.

Copyright Notice

Table of Contents

1. Introduction

Multicast technology is deployed more and more. Many modern technology use PIM technology include IPTV, Netmeeting, and so on. Except the unicast routes changing will cause the lost of multicast packets. The change of DR also cause the lost of multicast packets.

When the DR on a share-media LAN is down, other routers will elect the new DR until the expiration of Hello-Holdtime. The default value of Hello-Holdtime is 105 seconds. Although the value of Hello-Holdtime can be changed by manual, when the DR is down, there are many multicast packets will be lost. The quality of IPTV and Net Meeting will be influenced.

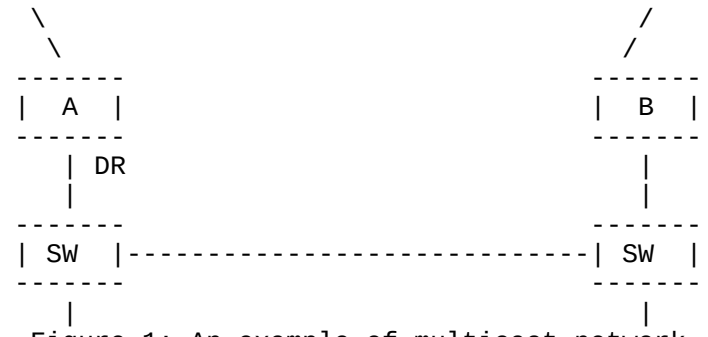


Figure 1: An example of multicast network

For example, there were two routers on one Ethernet. RouterA was elected to DR. When RouterA was down, the multicast packets are discarded until the RouterB was elected to DR and RouterB imported the multicast flows successfully.

We suppose there was only a RouterA in the Ethernet at first in Figure 1. RouterA was the DR who was responsible for forwarding multicast flows. When RouterB connected the Ethernet, RouterB would be elected to DR because a high priority. So RouterA stopped forwarding multicast packets. The multicast flows recovered until RouterB joined the group and imported multicast flows.

2. Terminology

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

3. PIM hello message format

In RFC4601, the PIM hello message format was defined. In this document, we define two new option values which are including Type, Length, and Value.

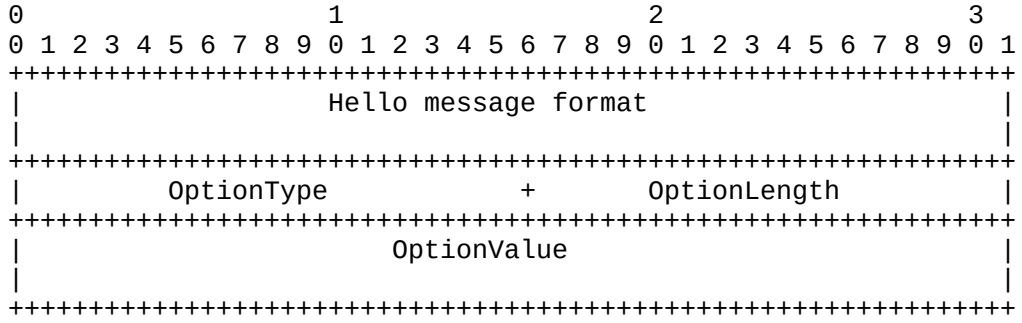


Figure 2: Hello message format

3.1. DR Option format

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

3.2. BDR Option format

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

4. The Protocol Treatment

A new router start sending hello messages with the values of DR and BDR are all set to 0 after its interface is enabled in PIM on a share-media LAN. When the router receive hello messages from other routers on the same share-media LAN, the router will check if the value of DR or BDR is filled. If the value of DR or BDR is filled with IP address of router who is sending hello messages, the router will store the IP address.

Then the router compare the priority and IP address itself to the stored IP address of DR and BDR accord to the algorithm of RFC 4601. If the router notice that it is better to be DR than the existed DR or BDR. The router will make itself the BDR, and send new hello messages with its IP address as BDR and existed DR. If the router notices that the existed DR is most priority in the share-media LAN, but the existed BDR is set to 0x0 in the received hello messages, or the existed BDR is not better than the new router to be DR except existed DR, the router will elect itself to BDR. If the router notices that it is not better to be DR than existed DR and BDR, the router will respect the PIM protocol.

When the new router become the new BDR, the router will join the existed multicast groups, import multicast flows from upstream routers. But the BDR MUST not forward the multicast flows to avoid the duplicate multicast packets in the share-media LAN. The router will monitor the DR. When the DR unavailable because of the down or other reasons. The BDR will forward multicast flows immediately.

4.1. Sending Hello Messages

When a new router’s interface is enabled in PIM protocol, the router send hello messages with the values of DR and BDR are filled with 0x0.

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

When a new router notices the existed DR and BDR is more priority than itself. The router will send hello messages with the values of DR and BDR are filled with existed DR and BDR.

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

4.2. Receiving Hello Messages

When the values of DR and BDR which are carried by hello messages are received is all set to 0x0, the router MUST elect the DR due to the algorithm of RFC4601. And elect a new BDR which are the best choice except DR.

When the value of DR which is carried by received hello messages is not 0x0, and the value of BDR is set to 0x0, the router will elect itself to BDR.

When the values of DR and BDR that carried by received hello messages all are not set to 0x0. The router will mark the existed DR, and compare it and the BDR in message. When the router notice it is better to be DR than existed BDR. The router will elect itself to the BDR.

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

4.3. The election of DR and BDR

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

When a new router start to work on a shared-media LAN and receive hello messages from other routers which are set DR value at least. The new router will not change the existed DR even if it is superior

to the existed DR. If the new router is superior to existed BDR, the new router will replace the place of BDR on the LAN.

When an existed router receives hello messages from a new router, and the existed router is DR on the LAN, the existed DR router will compare the new router and all the other routers on the LAN. If the new router is superior to the other entire router, the existed DR router will treat the new router as new BDR.

When an existed router receives hello messages from a new router, and the existed router is BDR on the LAN, the existed BDR will compare itself and the new router. If the new router is superior to itself, the existed BDR will elect the new router as new BDR, and set itself for nothing.

When an existed router receives hello messages from a new router, and the existed router is neither DR nor BDR on the LAN, the existed router will compare the existed BDR and the new router. If the new router is superior to the other entire router, the existed DR router will treat the new router as new BDR.

5. Deployment suggestion

If there are two and more routers on a share-media LAN, and the quality of LAN will be influenced by the lost of multicast packets, the LAN should deploy the function of DR and BDR in this document.

6. Security Considerations

For general PIM Security Considerations.

7. IANA Considerations

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

8. Normative References

[I-D.ietf-pim-rfc4601bis]
Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, J., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", draft-ietf-pim-rfc4601bis-05 (work in progress), May 2015.

[RFC4601] Fenner, B., Handley, M., Holbrook, H., and I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", RFC 4601, August 2006.

Authors' Addresses

Sandy Zhang
ZTE Corporation
No. 50 Software Ave, Yuhuatai Distinct
Nanjing 210000
China

Phone: +86-025-88014634
Email: zhang.zheng@zte.com.cn

Fangwei Hu
ZTE Corporation
No.889 Bibo Rd
Shanghai 201203
China

Phone: +86-21-68896273
Email: hu.fangwei@zte.com.cn

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

Email: xu.benchong@zte.com.cn