Internet Engineering Task Force PIM WG
INTERNET DRAFT
draft-ietf-pim-dm-new-v2-05.txt
Andrew Adams (NextHop Technolgies)
Jonathan Nicholas (ITT A/CD)
William Siadak (NextHop Technologies)

November 9, 2001

Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)

Status of this Document

This document is a product of the IETF PIM WG. Comments should be addressed to the authors, or the WG's mailing list at pim@catarina.usc.edu.

Abstract

This document specifies Protocol Independent Multicast - Dense Mode (PIM-DM). PIM-DM is a multicast routing protocol that uses the underlying unicast routing information base to flood multicast datagrams to all multicast routers. Prune messages are used to prevent future messages from propagating to routers with no group membership information.

Table of Contents

1. Introduction

This specification defines a multicast routing algorithm for multicast groups that are densely distributed across a network. This protocol does not have a topology discovery mechanism often used by a unicast routing protocol. It employs the same packet formats sparse mode PIM (PIM-SM) uses. This protocol is called PIM - Dense Mode. The foundation of this design was largely built on Deering's early work on IP multicast routing [1].

2. Terminology

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 and indicate requirement levels for compliant PIM-DM implementations.

3. Definitions

Multicast Routing Information Base (MRIB)

> This is the multicast topology table, which is typically derived from the unicast routing table, or routing protocols such as MBGP that carry multicast-specific topology information. PIM-DM uses the MRIB to make decisions regarding RPF interfaces.

Tree Information Base (TIB)

> This is the collection of state maintained by a PIM router and created by receiving PIM Join/Prune messages, PIM Assert messages, and IGMP information from local hosts. It essentially stores the state of all multicast distribution trees at that router.

Reverse Path Forwarding (RPF)

> RPF is a multicast forwarding mode where a data packet is accepted for forwarding if it is received on an interface used to reach the source in unicast.

Upstream Interface

> Interface towards the source of the datagram. Also known as the RPF Interface.

Downstream Interface

> All interfaces that are not upstream interfaces, including the router itself.

(S,G) Pair

> Source S and destination group G associated with an IP packet.

4. Pseudocode Notation

We use set notation in several places in this specification.

A (+) B

> is the union of two sets A and B.

A (-) B

> are the elements of set A that are not in set B.

NULL

> is the empty set or list.

Note that (+) and (-) operators are NOT commutative, and must be conducted in the order specified.

In addition we use C-like syntax:

= denotes assignment of a variable.

== denotes a comparison for equality.

!= denotes a comparison for inequality.

Braces { and } are used for grouping.

5. PIM-DM Protocol Overview

This section provides an overview of PIM-DM behavior. It is intended as an introduction to how PIM-DM works, and is NOT definitive. For the definitive specification, see Section 6. Protocol Specification.

PIM-DM assumes that when a source starts sending, all downstream systems want to receive multicast datagrams. Initially, multicast datagrams are flooded to all areas of the network. PIM-DM uses RPF to prevent looping of multicast datagrams while flooding. If some areas of the network do not have group members, PIM-DM will prune off the forwarding branch by instantiating prune state.

Prune state has a finite lifetime. When that lifetime expires, data will again be forwarded down the previously pruned branch.

Prune state is associated with an (S,G) pair. When a new member for a group G appears in a pruned area, a router can "graft" toward the source S for the group, thereby turning the pruned branch back into a forwarding branch.

The broadcast of datagrams followed by pruning of unwanted branches is often referred to as a flood and prune cycle and is typical of dense mode protocols.

In order to minimize repeated flooding of datagrams and subsequent pruning associated with a particular (S,G) pair, PIM-DM uses a state refresh message. This message is sent by the router(s) directly connected to the source and is propagated throughout the network. When received by a router on its RPF interface, the state refresh message causes an existing prune state to be refreshed.

Compared with multicast routing protocols with built in topology discovery mechanisms (e.g. DVMRP) PIM-DM has a simplified design and is not hard-wired into a specific topology discovery protocol. However, such a simplification does incur more overhead by causing flooding and pruning to occur on some links that could be avoided if sufficient topology information were available, i.e. to decide whether an interface leads to any downstream members of a particular group. Additional overhead is chosen in favor of the simplification and flexibility gained by not depending on a specific topology discovery protocol.

PIM-DM differs from PIM-SM in two essential ways: 1) There are no periodic joins transmitted, only explicitly triggered prunes and grafts. 2) There is no Rendezvous Point (RP). This is particularly important in networks that cannot tolerate a single point of failure. (An RP is the root of a shared multicast distribution tree. For more details see [3]).

6. Protocol Specification

The specification of PIM-DM is broken into several parts:

- Section 6.1 details the protocol state stored.
- Section 6.2 specifies the data packet forwarding rules.
- Section 6.3 specifies generation and processing of Hello messages.
- Section 6.4 specifies the Join, Prune and Graft generation and processing rules.
- Section 6.5 specifies the State Refresh generation and forwarding rules.
- Section 6.6 specifies the Assert generation and processing rules.
- Section 6.7 gives details on PIM-DM Packet Formats
- Section 6.8 summarizes PIM-DM timers and their defaults

6.1. PIM Protocol State

This section specifies all the protocol states that a PIM-DM implementation should maintain in order to function correctly. We term this state the Tree Information Base or TIB, as it holds the state of all the multicast distribution trees at this router. In this specification, we define PIM-DM mechanisms in terms of the TIB. However, only a very simple implementation would actually implement packet forwarding operations in terms of this state. Most implementations will use this state to build a multicast forwarding table, which would then be updated when the relevant state in the TIB changes.

Although we specify precisely the state to be kept, this does not mean that an implementation of PIM-DM needs to hold the state in this form. This is actually an abstract state definition, which is needed in order to specify the router's behavior. A PIM-DM implementation is free to hold whatever internal state it requires, and will still be conformant with this specification so long as it results in the same externally visible protocol behavior as an abstract router that holds the following state.

6.1.1. General Purpose State

A router stores the following non-group-specific state:

For each interface:

Neighbor State:

For each neighbor:

- Information from neighbor's Hello
- Neighbor's Gen ID.
- Neighbor's LAN Prune Delay
- Neighbor's Override Interval
- Neighbor's State Refresh Capability
- Neighbor liveness timer (NLT)

State Refresh Capable

6.1.2. (S,G) State

For every source/group pair (S,G), a router stores the following state:

(S,G) state:
For each interface:
Local Membership:
State: One of {"NoInfo", "Include"}

PIM (S,G) Prune State:

- State: One of {"NoInfo" (NI), "Pruned" (P), "PrunePending" (PP)}
- Prune Pending Timer (PPT)
- Prune Timer (PT)

(S,G) Assert Winner State:

- State: One of {"NoInfo" (NI), "I lost Assert" (L), "I won Assert" (W)}
- Assert Timer (AT)
- Assert winner's IP Address
- Assert winner's Assert Metric

Upstream interface-specific:
Graft/Prune State:

- State: One of {"NoInfo" (NI), "Pruned" (P), "Forwarding" (F), "AckPending" (AP) }
- GraftRetry Timer (GRT)
- Override Timer (OT)
- Prune Limit Timer (PLT)

Originator State

- Source Active Timer (SAT)
- State Refresh Timer (SRT)

6.1.3. State Summarization Macros

Using the state defined above, the following "macros" are defined and will be used in the descriptions of the state machines and pseudocode in the following sections.

The most important macros are those defining the outgoing interface list (or "olist") for the relevant state.

```
immediate_olist(S,G) = pim_nbrs (-) prunes(S,G)
                     (+) ( pim_include(*,G) (-) pim_exclude(S,G) )
                     (+) pim_include(S,G) (-) lost_assert(S,G) (-) boundary(G)
```

```
olist(S,G) = immediate_olist(S,G) (–) RPF_interface(S)
```

The macros `pim_include(*,G)` and `pim_include(S,G)` indicate the interfaces to which traffic might be forwarded or not forwarded because of hosts that are local members on those interfaces.

```
pim_include(*,G) =
   { all interfaces I such that:
```

```
         local_receiver_include(*,G,I)}

pim_include(S,G) =
   { all interfaces I such that:
      local_receiver_include(S,G,I)}

pim_exclude(S,G) =
   { all interfaces I such that:
      local_receiver_exclude(S,G,I) }
```

The macro `RPF_interface(S)` returns the RPF interface for source, S. That is to say, it returns the interface used to reach S as indicated by the MRIB.

The macro `local_receiver_include(S,G,I)` is true if the IGMP module or other local membership mechanism has determined that there are local members on interface I that desire to receive traffic sent specifically by S to G. The macro `local_receiver_include(*,G,I)` is true if the IGMP module or other local membership mechanism has determined that there are local members on interface I that desire to receive all traffic sent to G. Note that this determination is expected to account for membership joins initiated on or by the router. The macro `local_receiver_exclude(S,G,I)` is true if `local_receiver_include(*,G,I)` is true but none of the local members desire to receive traffic from S.

The set `pim_nbrs` is the set of all interfaces on which the router has at least one active PIM neighbor.

The set `prunes(S,G)` is the set of all interfaces on which the router has received Prune(S,G) messages:

```
prunes(S,G) =
   { all interfaces I such that DownstreamPState(S,G,I) is in Pruned state}
```

The set `lost_assert(S,G)` is the set of all interfaces on which the router has lost an (S,G) Assert.

```
lost_assert(S,G) =
   { all interfaces I such that lost_assert(S,G,I) == TRUE }

boundary(G) =
   { all interfaces I with an administratively scoped boundary for group G}
```

The following pseudocode macro definitions are also used in many places in the specification. Basically RPF' is the RPF neighbor towards a source unless a PIM-DM Assert has overridden the normal choice of neighbor.

```
neighbor RPF'(S,G) {
   if ( I_Am_Assert_loser(S, G, RPF_interface(S) )) {
      return AssertWinner(S, G, RPF_interface(S) )
   } else {
      return MRIB.next_hop( S )
   }
}
```

The macro `I_Am_Assert_loser(S, G, I)` is true if the Assert state machine (in XXX) for (S,G) on interface I is in the "I am Assert Loser" state.

6.2. Data Packet Forwarding Rules

The PIM-DM packet forwarding rules are defined below in pseudocode.

`iif` is the incoming interface of the packet.
`S` is the source address of the packet.
`G` is the destination address of the packet (group address).
`RPF_interface(S)` is the interface the MRIB indicates would be used to route packets to S.

First, an RPF check MUST be performed to determine whether the packet should be accepted based on TIB state and the interface on which that the packet arrived. Packets that fail the RPF check MUST NOT be forwarded and the router will conduct an assert process for the (S,G) pair specified in the packet. Packets for which a route to the source cannot be found MUST be discarded.

If the RPF check has been passed, an outgoing interface list is constructed for the packet. If this list is not empty, then the packet MUST be forwarded to all listed interfaces. If the list is empty, then the router will conduct a prune process for the (S,G) pair specified in the packet.

On receipt on a data packet from S addressed to G on interface iif:

```
if( iif == RPF_interface(S) AND UpstreamPState(S,G) != Pruned ) {
    oiflist = olist(S,G)
}
else {
    oiflist = NULL
}

forward packet on all interfaces in oiflist
```

This pseudocode employs the following "macro" definition:

`UpstreamPState(S,G)` is the state of the Upstream(S,G) state machine in 6.4.1. Upstream Prune, Join and Graft Messages.

6.3. Hello Messages

This section describes the generation and processing of Hello messages.

6.3.1. Sending Hello Messages

PIM-DM uses Hello messages to detect other PIM routers. Hello messages are sent periodically on each PIM enabled interface. Hello messages are multicast to address 224.0.0.13 (the ALL PIM ROUTERS group). When PIM is enabled on an interface or a router first starts, the Hello Timer (HT) MUST be set to a random value between 0 and Hello_Period. This prevents synchronization of Hello messages if multiple routers are powered on simultaneously.

After the initial Hello message, a Hello message MUST be sent every Hello_Period. A single Hello timer MAY be used to trigger sending Hello messages on all active interfaces. The Hello Timer SHOULD NOT be reset except when it expires.

6.3.2. Receiving Hello Messages

When a Hello message is received, the receiving router SHALL record the receiving interface and the sender. This information is retained for a number of seconds in the Hold Time field of the Hello Message. If a new Hello message is received from a particular neighbor N, the Neighbor Liveness Timer (NLT(N,I)) MUST be reset to the new value.

6.3.3. Hello Message Hold Time

The Hold Time in the Hello Message should be set to a value that can reasonably be expected to keep the Hello active until a new Hello message is received. On most links, this will be 3.5 times the value of Hello_Period.

If the Hold Time is set to '0xffff', the receiving router MUST NOT time out that Hello message. This feature might be used for on-demand links to avoid keeping the link up with periodic Hello messages.

If a Hold Time of '0' is received, the corresponding neighbor state is expired immediately. When an interface goes down or changes IP address, a Hello message with a zero Hold Time SHOULD be sent immediately (with the old IP address if the IP address is changed) to cause any PIM neighbors to remove the old information immediately.

6.3.4. Handling Router Failures

If a Hello message is received from an active downstream neighbor with a different Generation ID (GenID), the neighbor has restarted and may not contain the correct (S,G) state. The router MAY replay the last State Refresh message for any (S,G) pairs for which it is the Assert Winner indicating Prune and Assert status to the downstream router. These State Refresh messages SHOULD be sent out at t_override (see 6.8.1).

Upon startup, a router MAY use any State Refresh messages received within J/P_Override_Interval of its first Hello message on an interface to establish state information. The State Refresh source will be the RPF'(S), and Prune status for all interfaces will be set according to the Prune Indicator bit in the State Refresh message. If the Prune Indicator is set, the router will set the PruneLimitTimer to Prune_Holdtime and set the PruneTimer on all downstream interfaces to the State Refresh's Interval times two. The router will then propagate the State Refresh as described in section 6.5.1.

6.3.5. Reducing Prune Propagation Delay on LANs

If all routers on a LAN support the LAN Prune Delay option, then the PIM routers on that LAN SHOULD use the values received to adjust its J/P_Override_Interval on that interface. Briefly, to avoid synchronization of Prune Override (Join) messages when multiple downstream routers share a multi-access link, sending of such messages is delayed by a small random amount of time. The period of randomization is configurable and has a default value of 3 seconds.

Each router on the LAN expresses its view of the amount of randomization necessary in the Override Interval field of the LAN Prune Delay option. When all routers on a LAN use the LAN Prune Delay Option, all routers on the LAN SHOULD set their Override_Interval to the largest Override value on the LAN.

The LAN Delay inserted by a router in the LAN Prune Delay option expresses the expected message propagation delay on the link and SHOULD be configurable by the system administrator. When all routers on a link use the LAN Prune Delay Option, all routers on the LAN SHOULD set Propagation Delay to the largest LAN Delay on the LAN. PIM implementers should enforce a lower bound on the permitted values for this delay to allow for scheduling and processing delays within their router. Such delays may cause received messages to be processed later as well as triggered messages to be sent later than intended. Setting this LAN Prune Delay to too low a value may result in temporary forwarding outages because a downstream router will not be able to override a neighbor's prune message before the upstream neighbor stops forwarding.

6.4. PIM-DM Prune, Join and Graft Messages

This section describes the generation and processing of PIM-DM Join, Prune and Graft messages. Prune messages are sent towards an upstream neighbor for S to indicate that traffic from S addressed to group G is not desired. In the case of two downstream routers A and B, where A wishes to continue receiving data and B does not, A will send a Join in response to B's Prune to override the Prune. This is the only situation in PIM-DM in which a Join message is used.

Finally, a Graft message is used to re-join a previously pruned branch to the delivery tree.

6.4.1. Upstream Prune, Join and Graft Messages

The Upstream(S,G) state machine for sending Prune, Graft and Join messages is given below. There are three states.

Forwarding (F)

This is the starting state of the Upsteam(S,G) state machine. The state machine is in this state if it just started or if oiflist(S,G) != NULL.

Pruned(P)

The set, olist(S,G), is empty The router will not forward data from S addressed to group G.

AckPending(AP)

The router was in the Pruned(P) state but a transition has occurred in the Downstream(S,G) state machine for one of this (S,G) entry's outgoing interfaces indicating that traffic from S addressed to G should again be forwarded. A Graft message has been sent to RPF'(S) but a Graft Ack message has not yet been received.

In addition there are three state-machine-specific timers:

GraftRetry Timer (GRT(S,G))

This timer is set when a Graft is sent upstream. If a corresponding GraftAck is not received before the timer expires, then another Graft is sent and the GraftRetry Timer is reset. The timer is stopped when a Graft Ack message is received. This timer is normally set to Graft_Retry_Period (see 6.8.1).

Override Timer (OT(S,G))

This timer is set when a Prune(S,G) is received on the upstream interface where olist(S,G) != NULL. When the timer expires, a Join(S,G) message is sent on the upstream interface. This timer is normally set to t_override (see 6.8.1).

Prune Limit Timer (PLT(S,G))

This timer is used to rate-limit Prunes on a LAN. It is only used when the Upstream(S,G) state machine is in the Pruned state. A Prune cannot be sent if this timer is running. This timer is normally set to t_limit (see 6.8.1)

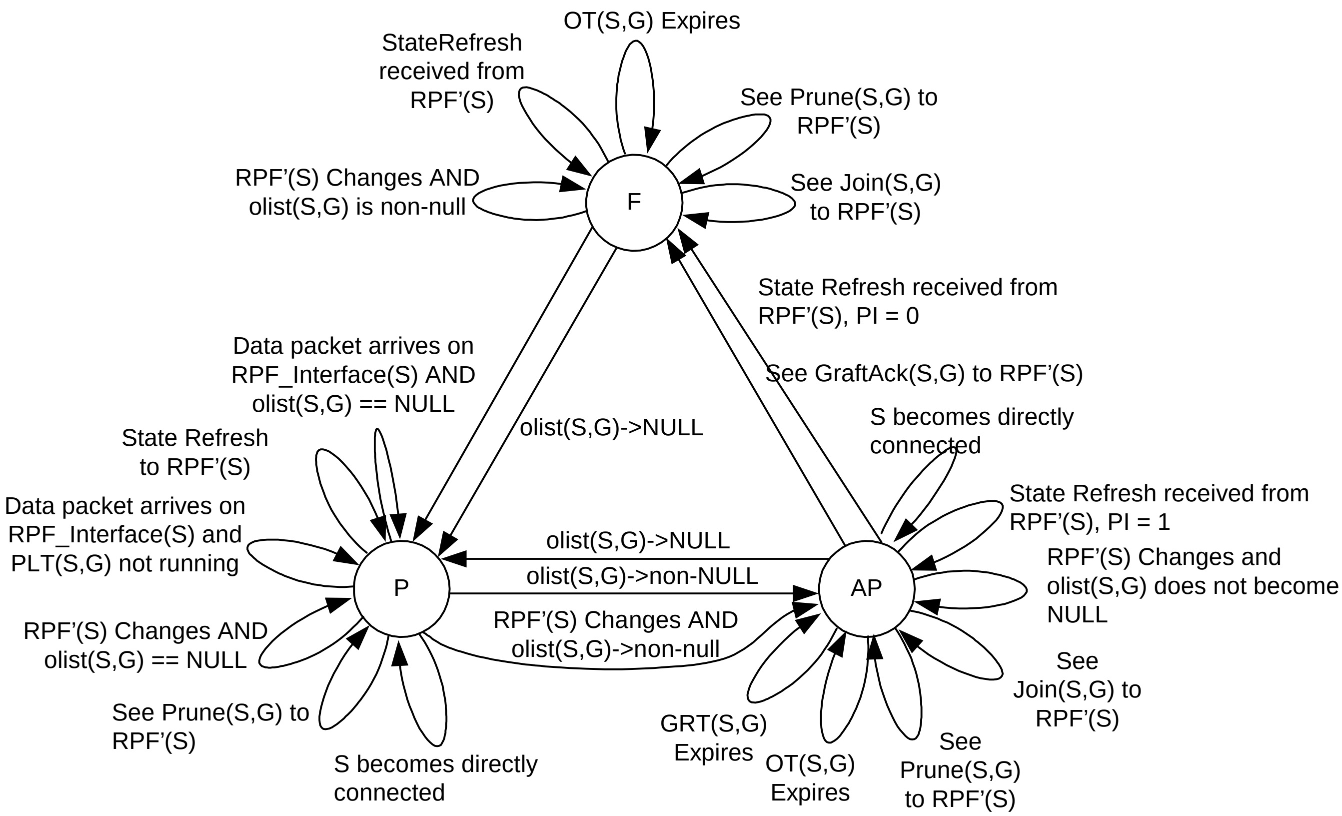

Figure 1 Upstream Interface State Machine

In tabular form, the state machine is defined as follows.

Event	Previous State: Forwarding	Pruned	AckPending
Data packet arrives on RPF_Interface(S) AND	->P Send Prune(S,G)	->P Send Prune(S,G)	N/A

olist(S,G) == NULL AND PLT(S,G) not running	Set PLT(S,G)	Set PLT(S,G)	
State Refresh(S,G) received from RPF’(S) AND Prune Indicator == 1	->F Set OT(S,G)	->P Reset PLT(S,G)	->AP Set OT(S,G)
State Refresh(S,G) received from RPF’(S) AND Prune Indicator == 0 AND PLT(S,G) not running	->F	->P Send Prune(S,G) Set PLT(S,G)	->F Cancel GRT(S,G)
See Join(S,G) to RPF’(S)	->F Cancel OT(S,G)	->P	->AP Cancel OT(S,G)
See Prune(S,G)	->F Set OT(S,G)	->P	->AP Set OT(S,G)
OT(S,G) Expires	->F Send Join(S,G)	N/A	->AP Send Join(S,G)
olist(S,G)->NULL	->P Send Prune(S,G) Set PLT(S,G)	N/A	->P Send Prune(S,G) Set PLT(S,G)
olist(S,G)->non-NULL	N/A	->AP Send Graft(S,G) Set GRT(S,G)	N/A
RPF’(S) Changes AND olist(S,G) != NULL	->AP Send Graft(S,G) Set GRT(S,G)	->AP Send Graft(S,G) Set GRT(S,G)	->AP Send Graft(S,G) Set GRT(S,G)
RPF’(S) Changes AND olist(S,G) == NULL	->P	->P Cancel PLT(S,G)	->P Cancel GRT(S,G)
S becomes directly connected	->F	->P	->F Cancel GRT(S,G)
GRT(S,G) Expires	N/A	N/A	->AP Send Graft(S,G) Set GRT(S,G)
Receive GraftAck(S,G) from RPF’(S)	->F	->P	->F Cancel GRT(S,G)

The transition event “RcvGraftAck(S,G)” implies receiving a Graft Ack message targeted to this router’s address on the incoming interface for the (S,G) entry. If the destination address is not correct, the state transitions in this state machine must not occur.

Transitions from the Forwarding (F) State

When the Upstream(S,G) state machine is in the Forwarding (F) state, the following events may trigger a transition:

Data Packet arrives on RPF_Interface(S) AND olist(S,G) == NULL AND S NOT directly connected

The Upstream(S,G) state machine MUST transition to the Pruned (P) state, send a Prune(S,G) to RPF'(S) and set PLT(S,G) to t_limit seconds.

olist(S,G) -> NULL AND S NOT directly connected

The Upstream(S,G) state machine MUST transition to the Pruned (P) state, send a Prune(S,G) to RPF'(S) and set PLT(S,G) to t_limit seconds.

See Prune(S,G) AND S NOT directly connected

This event is only relevant if RPF_interface(S) is a shared medium. This router sees another router on RPF_interface(S) send a Prune(S,G). As this router is in Forwarding state, it must override the Prune after a short random interval. If OT(S,G) is not running, the router MUST set OT(S,G) to t_override seconds.

The Upstream(S,G) state machine remains in Forwarding (F) state.

See Join(S,G) to RPF'(S)

This event is only relevant if RPF_interface(S) is a shared medium. This router sees another router on RPF_interface(S) send a Join(S,G) to RPF'(S,G). If the OT(S,G) is running, then it means that the router had scheduled a Join to override a previously received Prune. Another router has responded more quickly with a Join and so the local router SHOULD cancel its OT(S,G), if it is running.

The Upstream(S,G) state machine remains in the Forwarding (F) state.

OT(S,G) Expires AND S NOT directly connected

The OverrideTimer (OT(S,G)) expires. The router MUST send a Join(S,G) to RPF'(S) to override a previously detected prune.

The Upstream(S,G) state machine remains in the Forwarding (F) state.

RPF'(S) Changes AND olist(S,G) is non-null AND S NOT directly connected

Unicast routing or Assert state causes RPF'(S) to change, including changes to RPF_Interface(S).

The Upstream(S,G) state machine MUST transition to the AckPending (AP) state, unicast a Graft to the new RPF'(S) and set the GraftRetry Timer (GRT(S,G)) to Graft_Retry_Period.

State Refresh(S,G) Received from RPF'(S)

The Upstream(S,G) state machine remains in a Forwarding state.

If the received State Refresh has the Prune Indicator bit set to one, this router must override the upstream router's Prune state after a short random interval. If OT(S,G) is not running and the Prune Indicator bit equals one, the router MUST set OT(S,G) to t_override seconds.

Transitions from the Pruned (P) State

When the Upstream(S,G) state machine is in the Pruned (P) state, the following events may trigger a transition:

olist(S,G)->non-null AND S NOT directly connected
The set of interfaces defined by the olist(S,G) macro becomes non-empty indicating traffic from S addressed to group G must be forwarded.

The Upstream(S,G) state machine MUST cancel PLT(S,G), transition to the AckPending (AP) state and unicast a Graft message to RPF'(S). The Graft Retry Timer (GRT(S,G)) MUST be set to Graft_Retry_Period.

See Prune(S,G) to RPF'(S)
A Prune(S,G) is seen on `RPF_interface(S)` to RPF'(S).

The Upstream(S,G) state machine stays in the Pruned (P) state. The router MAY reset its PLT(S,G) to the value in the Holdtime field of the received message if greater than the current value of the PLT(S,G).

RPF'(S) Changes AND olist(S,G) -> non-null AND S NOT directly connected
Unicast routing or Assert state causes RPF'(S) to change, including changes to RPF_Interface(S).

The Upstream(S,G) state machine MUST cancel PLT(S,G), transition to the AckPending (AP) state, send a Graft unicast to the new RPF'(S) and set the GraftRetry Timer (GRT(S,G)) to Graft_Retry_Period.

RPF'(S) Changes AND olist(S,G) == NULL AND S NOT directly connected
Unicast routing or Assert state causes RPF'(S) to change, including changes to RPF_Interface(S).

The Upstream(S,G) state machine stays in the Pruned (P) state and MUST cancel the PLT(S,G) timer.

S becomes directly connected

Unicast routing changed so that S is directly connected. The Upstream(S,G) state machine remains in the Pruned (P) state.

Data arrives on RPF_interface(S) AND PLT(S,G) not running AND S NOT directly connected
Either another router on the LAN desires traffic from S addressed to G or a previous Prune was lost. In order to prevent generating a Prune(S,G) in response to every data packet, the PruneLimit Timer (PLT(S,G)) is used. Once the PLT(S,G) expires, the router needs to send another prune in response to a data packet not received directly from the source.

A Prune(S,G) MUST be sent to RPF'(S) and the PLT(S,G) MUST be set to t_limit.

State Refresh(S,G) Received from RPF'(S)

The Upstream(S,G) state machine remains in a Pruned state.

If the State Refresh has its Prune Indicated bit set to zero and PLT(S,G) is not running, a Prune(S,G) MUST be sent to RPF'(S) and the PLT(S,G) MUST be set to t_limit. If the State Refresh has its Prune Indicated bit set to one, the router MUST reset PLT(S,G) to t_limit.

Transitions from the AckPending (AP) State

When the Upstream(S,G) state machine is in the AckPending (AP) state, the following events may trigger a transition:

GRT(S,G) Expires

The GraftRetry Timer (GRT(S,G)) expires for this (S,G) entry.

The Upstream(S,G) state machine stays in the AckPending (AP) state. Another Graft message for (S,G) SHOULD be unicasted to RPF'(S) and the GraftRetry Timer (GRT(S,G)) reset to Graft_Retry_Period. Note that RPF'(S) may have changed since the previous Graft.

RPF'(S) Changes AND olist(S,G) does not become NULL

Unicast routing or Assert state causes RPF'(S) to change, including changes to RPF_Interface(S).

The Upstream(S,G) state machine stays in the AckPending (AP) state. A Graft MUST be unicast to the new RPF'(S) and the GraftRetry Timer (GRT(S,G)) reset to Graft_Retry_Period.

S becomes directly connected

Unicast routing has changed so that S is directly connected. The GraftRetry Timer MUST be cancelled and the Upstream(S,G) state machine MUST transition to the Forwarding(F) state.

See GraftAck(S,G) from RPF'(S)

A GraftAck is received from RPF'(S).

The GraftRetry Timer MUST be cancelled and the Upstream(S,G) state machine MUST transition to the Forwarding(F) state.

olist(S,G) -> NULL

The set of interfaces defined by the `olist(S,G)` macro becomes null indicating traffic from S addressed to group G should no longer be forwarded.

The Upstream(S,G) state machine MUST transition to the Pruned (P) state. A Prune(S,G) MUST be multicast to the `RPF_interface(S)` with RPF'(S) named in the upstream neighbor field. The GraftRetry Timer (GRT(S,G)) MUST be cancelled.

See Prune(S,G)

This event is only relevant if RPF_interface(S) is a shared medium. This router sees another router on RPF_interface(S) send a Prune(S,G). As this router is in AckPending (AP) state, it must override the Prune after a short random interval. If OT(S,G) is not running, the router MUST set OT(S,G) to t_override seconds.

The Upstream(S,G) state machine remains in AckPending (AP) state.

See Join(S,G) to RPF'(S,G)

This event is only relevant if RPF_interface(S) is a shared medium. This router sees another router on RPF_interface(S) send a Join(S,G) to RPF'(S,G). If the OT(S,G) is running, then it means that the router had scheduled a Join to override a previously received Prune. Another router has responded more quickly with a Join and so the local router SHOULD cancel its OT(S,G), if it is running.

The Upstream(S,G) state machine remains in the AckPending (AP) state.

OT(S,G) Expires

The OverrideTimer (OT(S,G)) expires. The router MUST send a Join(S,G) to RPF'(S).

The Upstream(S,G) state machine remains in the AckPending (AP) state.

State Refresh(S,G) Received from RPF'(S) with Prune Indicator == 1

The Upstream(S,G) state machine remains in an AckPending state.

The router must override the upstream router's Prune state after a short random interval. If OT(S,G) is not running and the Prune Indicator bit equals one, the router MUST set OT(S,G) to t_override seconds.

State Refresh(S,G) Received from RPF'(S) with Prune Indicator == 0

The router MUST cancel its GraftRetry Timer (GRT(S,G)) and transition to the Forwarding (F) state.

6.4.2. Downstream Prune, Join and Graft Messages

The Prune(S,G) Downstream state machine for receiving Prune, Join and Graft messages on interface I is given below. This state machine MUST always be in the NoInfo state on the upstream interface. It contains three states.

NoInfo(NI)

The interface has no (S,G) Prune state and neither the Prune timer (PT(S,G,I)) nor the PrunePending timer ((PPT(S,G,I)) is running.

PrunePending(PP)

The router has received a Prune(S,G) 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 PrunePending state functions exactly like the NoInfo state.

Pruned(P)

The router has received a Prune(S,G) on this interface from a downstream neighbor and the Prune was not overridden. Data from S addressed to group G is no longer being forwarded on this interface.

In addition there are two timers:

PrunePending Timer (PPT(S,G,I))

This timer is set when a valid Prune(S,G) is received. Expiry of the PrunePending Timer (PPT(S,G,I)) causes the interface to transition to the Pruned state.

Prune Timer (PT(S,G,I))

This timer is set when the PrunePending Timer (PT(S,G,I)) expires. Expiry of the Prune Timer (PT(S,G,I)) causes the interface to transition to the NoInfo (NI) state, thereby allowing data from S addressed to group G to be forwarded on the interface.

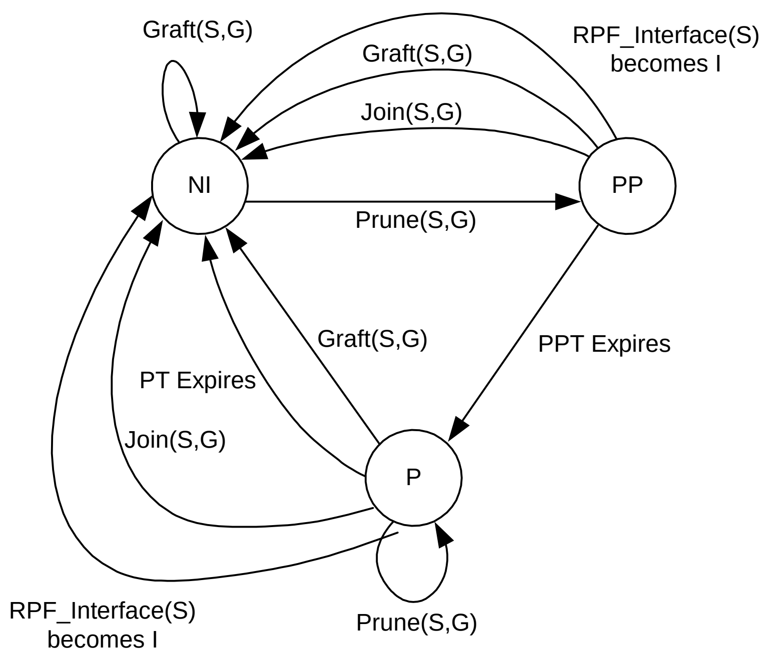

Figure 2: Prune(S,G) Downstream State Machine

In tabular form, the state machine is:

```
+-------------------------------+------------------------------------+
|                               |           Previous State           |
+                               +------------+------------+------------+
|            Event              |  No Info   | PrunePend  |   Pruned   |
+-------------------------------+------------+------------+------------+
| Receive Prune(S,G)            |->PP  Set   |->PP        |->P Reset   |
|                               | PPT(S,G,I) |            |  PT(S,G,I) |
+-------------------------------+------------------------------------+
| Receive Join(S,G)             |->NI        |->NI Cancel |->NI Cancel |
|                               |            | PPT(S,G,I) |  PT(S,G,I) |
```

Receive Graft(S,G)	->NI Send GraftAck	->NI Send GraftAck Cancel PPT(S,G,I)	->NI Send GraftAck Cancel PT(S,G,I)
PPT(S,G) Expires	N/A	->P Set PT(S,G,I)	N/A
PT(S,G) Expires	N/A	N/A	->NI
RPF_Interface(S) becomes I	->NI	->NI Cancel PPT(S,G,I)	->NI Cancel PT(S,G,I)

The transition events “Receive Graft(S,G)”, “Receive Prune(S,G)” and “Receive Join(S,G)” denote receiving a Graft, Prune or Join message in which this router’s address on I is contained in the message’s upstream neighbor field. If the upstream neighbor field does not match this router’s address on I, then these state transitions in this state machine must not occur.

Transitions from the NoInfo State

When the Prune(S,G) Downstream state machine is in the NoInfo (NI) state, the following events may trigger a transition:

Receive Prune(S,G)

A Prune(S,G) is received on interface I with the upstream neighbor field set to the router’s address on I.

The Prune(S,G) Downstream state machine on interface I MUST transition to the PrunePending (PP) state. The PrunePending Timer (PPT(S,G,I)) MUST be set to J/P_Override_Interval if the router has more than one neighbor on I. If the router has only one neighbor on interface I, then it SHOULD set the PPT(S,G,I) to zero, effectively transitioning immediately to the Pruned (P) state.

Receive Graft(S,G)

A Graft(S,G) is received on the interface I with the upstream neighbor field set to the router’s address on I.

The Prune(S,G) Downstream state machine on interface I stays in the NoInfo (NI) state. A GraftAck(S,G) MUST be unicasted to the originator of the Graft(S,G) message.

Transitions from the PrunePending (PP) State

When the Prune(S,G) downstream state machine is in the PrunePending (PP) state, the following events may trigger a transition.

Receive Graft(S,G)

A Graft(S,G) is received on interface I with the upstream neighbor field set to the router’s address on I.

The Prune(S,G) Downstream state machine on interface I MUST transition to the NoInfo (NI) state and MUST unicast a Graft Ack message to the Graft originator. The PrunePending Timer (PPT(S,G,I)) MUST be cancelled.

Receive Join(S,G)

A Join(S,G) is received on interface I with the upstream neighbor field set to the router's address on I.

The Prune(S,G) Downstream state machine on interface I MUST transition to the NoInfo (NI) state. The PrunePending Timer (PPT(S,G,I)) MUST be cancelled.

PPT(S,G,I) Expires

The PrunePending Timer (PPT(S,G,I)) expires indicating that no neighbors have overridden the previous Prune(S,G) message.

The Prune(S,G) Downstream state machine on interface I MUST transition to the Pruned (P) state.. The Prune Timer (PT(S,G,I)) is started and MUST be initialized to the received Prune_Hold_Time minus J/P_Override_Interval. A PruneEcho(S,G) MUST be sent on I if I has more than one PIM neighbor.

A PruneEcho(S,G) is simply a Prune(S,G) message multicast by the upstream router to a LAN with itself as the Upstream Neighbor. Its purpose is to add additional reliability so that if a Join that should have overridden the Prune is lost locally on the LAN, then the PruneEcho(S,G) may be received and trigger a new Join message . A PruneEcho(S,G) is OPTIONAL on an interface with only one PIM neighbor.

RPF_Interface(S) becomes interface I

The upstream interface for S has changed.

The Prune(S,G) Downstream state machine on interface I MUST transition to the NoInfo (NI) state. The PrunePending Timer (PPT(S,G,I)) MUST be cancelled.

Transitions from the Prune (P) State

When the Prune(S,G) Downstream state machine is in the Pruned (P) state, the following events may trigger a transition.

Receive Graft(S,G)

A Graft(S,G) is received on interface I with the upstream neighbor field set to the router's address on I.

The Prune(S,G) Downstream state machine on interface I MUST transition to the NoInfo (NI) state. and send a Graft Ack back to the Graft's source. The Prune Timer (PT(S,G,I)) MUST be cancelled. The router MUST evaluate any possible transitions in the Upstream(S,G) state machine.

Receive Join(S,G)

A Join(S,G) is received on the interface I with the upstream neighbor field set to the router's address on I.

The Prune(S,G) downstream state machine on interface I MUST transition to the NoInfo (NI) state. The Prune Timer (PT(S,G,I)) MUST be cancelled. The router MUST evaluate any possible transitions in the Upstream(S,G) state machine.

Receive Prune(S,G)

A Prune(S,G) is received on the interface I with the upstream neighbor field set to the router's address on I.

The Prune(S,G) Downstream state machine on interface I remains in the Pruned (P) state. The Prune Timer (PT(S,G,I)) SHOULD be reset to the holdtime contained in the Prune(S,G) message.

PT(S,G,I) Expires

The Prune Timer (PT(S,G,I)) expires indicating that it is again time to flood data from S addressed to group G onto interface I.

The Prune(S,G) Downstream state machine on interface I MUST transition to the NoInfo (NI) state. The router MUST evaluate any possible transitions in the Upstream(S,G) state machine.

RPF_Interface(S) becomes interface I

The upstream interface for S has changed.

The Prune(S,G) Downstream state machine on interface I MUST transition to the NoInfo (NI) state. The PruneTimer (PT(S,G,I)) MUST be cancelled.

6.5. State Refresh

This section describes the major portions of the state refresh mechanism.

6.5.1. Forwarding of State Refresh Messages

When a State Refresh message, SRM, is received, it is forwarded according to the following pseudo-code.

```
If (iif != RPF_interface(S))
      return;
if (RPF'(S) != srcaddr(SRM))
      return;
for each interface I in pim_nbrs {
      if (TTL(SRM) == 0 OR TTL(SRM) –1 < Threshold(I))
            continue;     /* Out of TTL, skip this interface */
      if (boundary(I,G))
            continue;     /* This interface is scope boundary, skip it */
      if (I == iif)
            continue;     /* This is the incoming interface, skip it */
      Copy SRM to SRM';   /* Make a copy of SRM to forward */
      if (I contained in Prunes(S,G)) {
            set Prune Indicator bit of SRM' to 1;
            if StateRefreshCapable(I) == TRUE
                  set PT(S,G) to largest active holdtime read from a Prune
message accepted on I;
      }
```

```
        else {
              set Prune Indicator bit of SRM’ to 0;
        }
        set srcaddr(SRM’) to my_addr(I);
        set TTL of SRM’ to TTL(SRM) – 1;
        set metric of SRM’ to metric of unicast route used to reach S;
        set pref of SRM’ to prference of unicast route used to reach S;
        set mask of SRM’ to mask of route used to reach S;
        if (AssertState == NoInfo) {
              set Assert Override of SRM’ to 1;
        }
        else {
              set Assert Override of SRM’ to 0;
        }     transmit SRM’ on I;
}
```

The pseudocode above employs the following macro definitions.

`Boundary(I,G)` evaluates to TRUE if an administratively scoped boundary for group G is configured on interface I.

`StateRefreshCapable(I)` evaluates to TRUE if all neighbors on an interface use the State Refresh option.

`TTL(SRM)` returns the TTL contained in the State Refresh Message, SRM. This is different from the TTL contained in the IP header.

`Threshold(I)` returns the minimum TTL that a packet must have before it can be transmitted on interface I.

`srcaddr(SRM)` returns the source address contained in the network protocol (e.g. IPv4) header of the State Refresh Message, SRM.

`my_addr(I)` returns this node’s network (e.g. IPv4) address of interface I.

6.5.2. State Refresh Message Origination

This section describes the origination of State Refresh messages. These messages are generated periodically by the PIM-DM router that is directly connected to a source. One Origination(S,G) state machine exists per (S,G) entry in a PIM-DM router.

The Origination(S,G) state machine has the following states.

NotOriginator(NO)

This is the starting state of the Origination(S,G) state machine. While in this state a router will not originate State Refresh messages for the (S,G) pair.

Originator(O)

When in this state the router will periodically originate State Refresh messages. Only routers which are directly connected to S may transition to this state.

In addition there are two state-machine-specific timers:

StateRefresh Timer (SRT(S,G))

This timer is controls when State Refresh messages are generated. The timer is initially set when that Origination(S,G) state machine transitions to the O state. It is cancelled when the Origination(S,G) state machine transitions to the NO state. This timer is normally set to StateRefreshInterval (see 6.8.1).

SourceActive Timer (SAT(S,G))

This timer is first set when the Origination(S,G) state machine transitions to the O state and is reset on the receipt of every data packet from S addressed to group G. When it expires, the Origination(S,G) state machine transitions to the NO state. This timer is normally set to SourceLifetime (see 6.8.1).

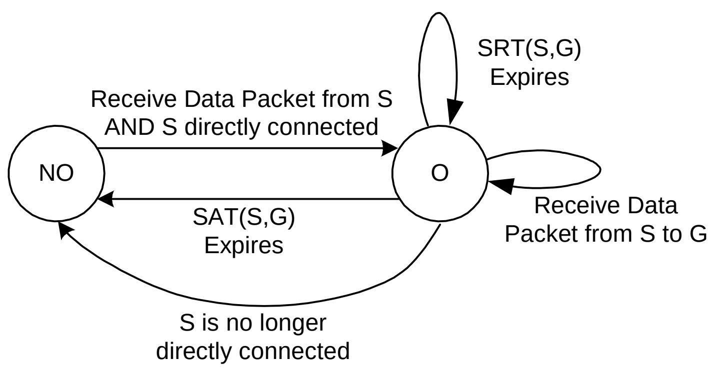

Figure 3 Per-interface State Refresh State Diagram

In tabular form, the state machine is defined as follows.

In NotOriginator (NO) State
Receive Data Packet from S AND S directly connected
->O state

In Originator (O) State			
SRT(S,G) Expires	**SAT(S,G) Expires**	**Receive Data Packet from S to G**	**S no longer directly connected**
->O state	-> NO state	-> O state	-> NO state

Transitions from the NotOriginator (NO) State

When the Originating(S,G) state machine is in the NotOriginator (NO) state, the following event may trigger a transition:

Data Packet received from directly connected Source S addressed to group G

The router MUST transition to an Originator (O) state, set SAT(S,G) to SourceLifetime, and set SRT(S,G) to StateRefreshInterval. The router SHOULD record the TTL of the packet for use in State Refresh messages.

Transitions from the Originator (O) State

When the Originating(S,G) state machine is in the Originator (O) state, the following events may trigger a transition:

SRT(S,G) Expires

The router remains in the Originator (O) state and resets SRT(S,G) to StateRefreshInterval. The router also generates State Refresh messages for transmission over each interface on which there are PIM-DM neighbors except for the interface by which S is reached. If the TTL of data packets from S to G are being recorded, then the TTL of each State Refresh message is set to the highest recorded TTL. Otherwise, the TTL is set to the TTL of the interface over which the State Refresh message will be sent.

Let I denote the interface over which a State Refresh message is being sent. If the Prune(S,G) Downstream state machine for I is in the NoInfo (NI) state, then the Prune-Indicator bit should be set to 0 in the State Refresh message being sent over I. Otherwise the Prune-Indicator bit should be set to 1.

SAT(S,G) Expires

The router cancels the SRT(S,G) timer and transitions to the NotOriginator (NO) state.

Receive Data Packet from S addressed to G

The router remains in the Originator (O) state and resets SAT(S,G) to SourceLifetime. The router SHOULD increase its recorded TTL to match the TTL of the packet, if the packet's TTL is larger than the previously recorded TTL.

S is no longer directly connected

The router remains transitions to the NotOriginator (NO) state and cancels both the SAT(S,G) and SRT(S,G).6.6. PIM Assert Messages

6.6.1. Assert Metrics

Assert metrics are defined as:

```
struct assert_metric {
   metric_preference;
   route_metric;
   ip_address;
};
```

When comparing assert_metrics, the metric_preference and route_metric field are compared in order, where the first lower value wins. If all fields are equal, the IP address of the router that sourced the Assert message is used as a tie-breaker, with the highest IP address winning.

An Assert metric for (S,G) to include in (or compare against) an Assert message sent on interface I should be computed using the following pseudocode:

```
assert_metric
my_assert_metric(S,G,I) {
   if( CouldAssert(S,G,I) == TRUE ) {
      return spt_assert_metric(S,G,I)
   else {
      return infinite_assert_metric()
   }
}
```

`spt_assert_metric(S,I)` gives the Assert metric we use if we're sending an Assert based on active (S,G) forwarding state:

```
      assert_metric
      spt_assert_metric(S,I) {
         return {0,MRIB.pref(S),MRIB.metric(S),my_ip_address(I)}
}
```

`MRIB.pref(X)` and `MRIB.metric(X)` are the routing preference and routing metrics associated with the route to a particular (unicast) destination X, as determined by the MRIB. `my_ip_address(I)` is simply the router's IP address that is associated with the local interface I.

`infinite_assert_metric()` gives the Assert metric we need to send an Assert but doesn't match (S,G) forwarding state:

```
      assert_metric
         infinite_assert_metric() {
            return {1,infinity,infinity,infinity}
}
```

6.6.2. AssertCancel Messages

An AssertCancel(S,G) message is simply an Assert message for (S,G) with infinite metric. The Assert winner sends such a message when it changes its upstream interface to this interface. Other routers will see this metric, causing those with forwarding state to send their own Asserts and re-establish an Assert winner.

AssertCancel messages are simply an optimization. The original Assert timeout mechanism will allow a subnet to eventually become consistent; the AssertCancel mechanism simply causes faster convergence. No special processing is required for an AssertCancel message, since it is simply an Assert message from the current winner.

6.6.3. Assert State Macros

The macro `lost_assert(S,G,I)`, is used in the olist computations of "6.1.3. State Summarization Macros", and is defined as follows.

```
bool lost_assert(S,G,I) {
   if ( RPF_interface(S) == I ) {
      return FALSE
   } else {
      return ( AssertWinner(S,G,I) != me  AND
               (AssertWinnerMetric(S,G,I) is better
```

```
                      than spt_assert_metric(S,G,I) )
         }
      }
```
`AssertWinner(S,G,I)` defaults to Null and `AssertWinnerMetric(S,G,I)` defaults to Infinity when in the NoInfo state.

6.6.4. (S,G) Assert Message State Machine

The (S,G) Assert state machine for interface I is shown in Figure 3. There are three states:

NoInfo (NI)

This router has no (S,G) Assert state on interface I.

I am Assert Winner (W)

This router has won an (S,G) Assert on interface I. It is now responsible for forwarding traffic from S destined for G via interface I.

I am Assert Loser (L)

This router has lost an (S,G) Assert on interface I. It must not forward packets from S destined for G onto interface I.

In addition there is also an Assert Timer (AT(S,G,I)) that is used to time out Assert state on the Assert losers and to resend Assert messages on the Assert winner.

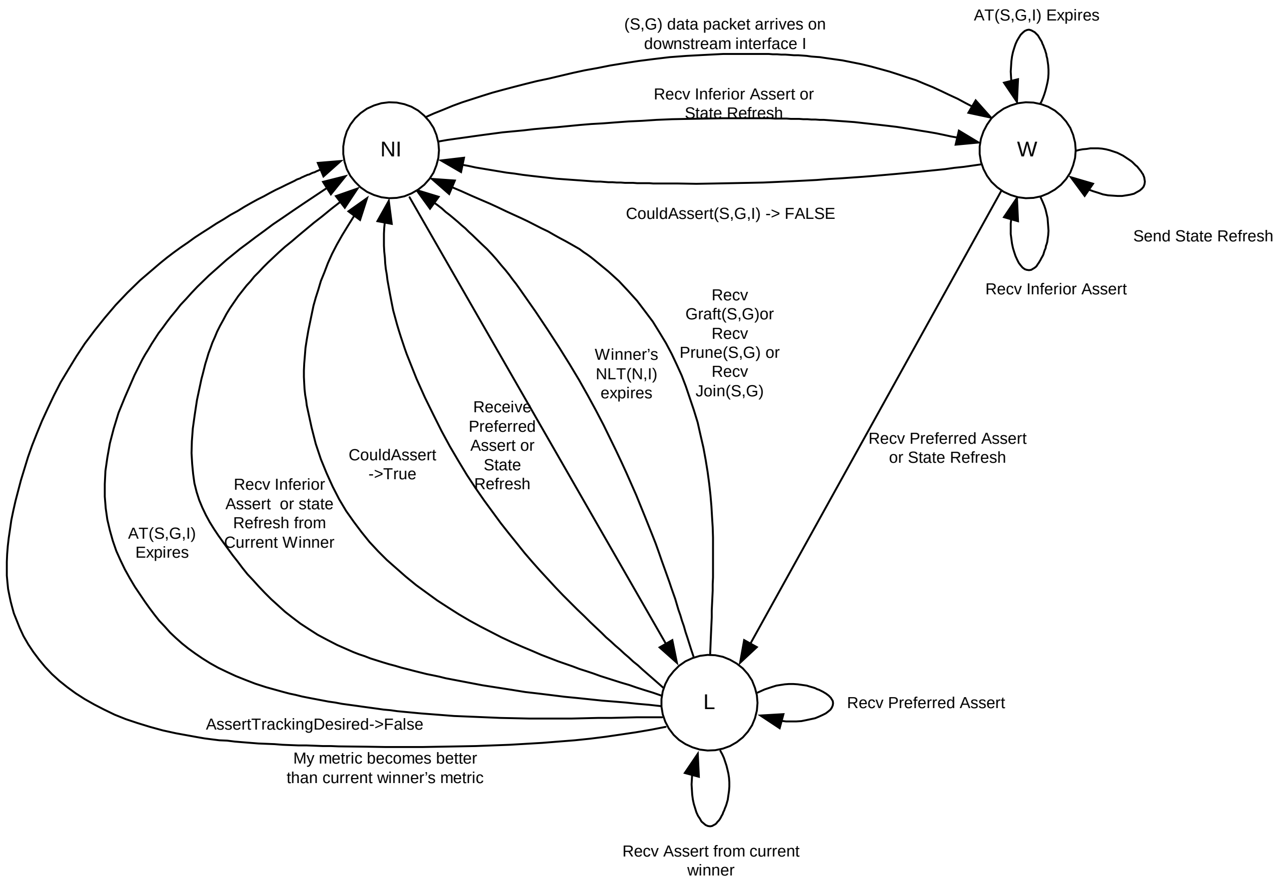

Figure 4: Per-interface (S,G) Assert state machine

In tabular form the state machine is defined as follows.

	Previous State		
Event	No Info	Winner	Loser
An (S,G) Data packet received on downstream interface	->W Send Assert(S,G) Set AT(S,G,I)	->W Send Assert(S,G) Set AT(S,G,I)	->L
Receive Inferior (Assert OR State Refresh) from Assert Winner	N/A	N/A	->NI Cancel AT(S,G,I)
Receive Inferior (Assert OR State Refresh) from non-Assert Winner AND CouldAssert==TRUE	->W Send Assert(S,G) Set AT(S,G,I)	->W Send Assert(S,G) Set AT(S,G,I)	->L

Receive Preferred Assert OR State Refresh	->L Send Prune(S,G) Set AT(S,G,I)	->L Send Prune(S,G) Set AT(S,G,I)	->L Set AT(S,G,I)
Send State Refresh	->NI	->W Reset AT(S,G,I)	->L
AT(S,G) Expires	N/A	->NI	->NI
CouldAssert -> FALSE	->NI	->NI Cancel AT(S,G,I)	->NI Cancel AT(S,G,I)
CouldAssert -> TRUE	->NI	N/A	->NI Cancel AT(S,G,I)
Winner’s NLT(N,I) Expires	N/A	N/A	->NI Cancel AT(S,G,I)
Receive Prune(S,G), Join(S,G) or Graft(S,G)	->NI	->W	->L Send Assert(S,G)

Terminology:

A "preferred assert" is one with a better metric than the current winner.

An "inferior assert" is one with a worse metric than `my_assert_metric(S,G,I)`.

The state machine uses the following macros:

```
CouldAssert(S,G,I) = (RPF_interface(S) != I)
```

The first line accounts for (S,G) join information received on I that might cause the router to be interested in Asserts on I.

The last line accounts for the fact that a router must keep track of Assert information on upstream interfaces in order to send Grafts and Prunes to the proper neighbor.

Transitions from NoInfo State

When in NoInfo state, the following events may trigger transitions:

Receive Inferior (Assert OR State Refresh) AND CouldAssert(S,G,I)==TRUE

An Assert or State Refresh is received for (S,G) that is inferior to our own assert metric on interface I. The Assert state machine MUST transition to the "I am Assert Winner" state, send an Assert(S,G) to interface I, store its own address and metric as the Assert Winner and set the Assert Timer (AT(S,G,I)) to (Assert_Time – Assert_Override_Interval).

An (S,G) data packet arrives on downstream interface I

An (S,G) data packet arrived on a downstream interface which is contained in immediate_olist(S,G). It is optimistically assumed that this router will be the Assert winner for this (S,G). The Assert state machine MUST transition to the "I am Assert Winner" state, send an Assert(S,G) to interface I, store its own address and metric as the Assert Winner and set the Assert_Timer (AT(S,G,I) to (Assert_Time – Assert_Override_Interval), thereby initiating the Assert negotiation for (S,G).

Receive Preferred Assert or State Refresh

The received Assert or State Refresh has a better metric than this router's and therefore the Assert state machine MUST transition to the "I am Assert Loser" state and store the Assert Winner's address and metric. If the metric was received in an Assert, the router MUST set the Assert Timer (AT(S,G,I)) to Assert_Time. If the metric was received in a State Refresh, the router MUST set the Assert Timer (AT(S,G,I)) to three times the received State Refresh Interval. The router MUST also multicast a Prune(S,G) to the Assert winner and evaluate any changes in its Upstream(S,G) state machine.

Transitions from Winner State

When in "I am Assert Winner" state, the following events trigger transitions:

AT(S,G,I) Expires

The (S,G) Assert Timer (AT(S,G,I)) expires. The Assert state machine MUST transition to the NoInfo (NI) state.

Send State Refresh

The router is sending a State Refresh(S,G) message on interface I. The router MUST set the Assert Timer (AT(S,G,I)) to three times the State Refresh Interval contained in the State Refresh(S,G) message.

Receive Inferior Assert

An (S,G) Assert is received containing a metric for S that is worse metric than this router's metric for S. Whoever sent the Assert is in error. The router MUST send an Assert(S,G) to interface I and reset the Assert Timer (AT(S,G,I)) to (Assert_Time – Assert_Override_Interval).

Receive Preferred Assert or State Refresh

An (S,G) Assert or State Refresh is received that has a better metric than this router's metric for S on interface I. The Assert state machine MUST transition to "I am Assert Loser" state and store the Assert Winner's address and metric. If the metric was received in an Assert, the router MUST set the Assert Timer (AT(S,G,I)) to Assert_Time. If the metric was received in a State Refresh, the router MUST set the Assert Timer (AT(S,G,I)) to three times the State Refresh Interval. The router MUST also multicast a Prune(S,G) to the Assert winner and evaluate any changes in its Upstream(S,G) state machine.

CouldAssert(S,G,I) -> FALSE

This router's RPF interface changed so as to make `CouldAssert(S,G,I)` become false. This router can no longer perform the actions of the Assert winner, and so the Assert state machine MUST transition to NoInfo (NI) state, send an AssertCancel(S,G) to interface I, and remove itself as the Assert Winner.

Transitions from Loser State

When in "I am Assert Loser" state, the following transitions can occur:

Receive Preferred Assert or State Refresh

An Assert or State Refresh is received that has a better metric than that of the current Assert winner. The Assert state machine remains in Loser (L) state and MUST store the address and metric of the new Assert Winner. If the metric was received in an Assert, the router MUST set the Assert Timer (AT(S,G,I)) to Assert_Time. If the metric was received in a State Refresh, the router MUST set the Assert Timer (AT(S,G,I)) to three times the received State Refresh Interval.. If CouldAssert == TRUE, the router MUST multicast a Prune(S,G) to the new Assert winner.

CouldAssert -> TRUE

CouldAssert has become TRUE because interface I used to be the RPF interface for S, and now it is not. The Assert state machine MUST transition to NoInfo (NI) state, cancel AT(S,G,I) and delete information concerning the Assert Winner on I.

Receive Inferior Assert or State Refresh from Current Winner

An Assert or State Refresh is received from the current Assert winner that is worse than this router's metric for S (typically the winner's metric became worse). The Assert state machine MUST transition to NoInfo (NI) state and cancel AT(S,G,I). The router MUST delete the previous Assert Winner's address and metric and evaluate any possible transitions to its Upstream(S,G) state machine. Usually this router will eventually re-assert and win when data packets from S have started flowing again.

AT(S,G,I) Expires

The (S,G) Assert Timer (AT(S,G,I)) expires. The Assert state machine MUST transition to NoInfo (NI) state. The router MUST delete the Assert Winner's address and metric. If CouldAssert == TRUE, the router MUST evaluate any possible transitions to its Upstream(S,G) state machine.

My metric becomes better than the assert winner's metric AND CouldAssert == TRUE

`my_assert_metric(S,G,I)` has changed so that now this router's Assert metric for (S,G) is better than the metric it has stored for current Assert winner. This might happen when the underlying routing metric changes, or when `CouldAssert(S,G,I)` becomes true. The Assert state machine MUST transition to NoInfo (NI) state, delete the Assert Winner's address and metric, and evaluate any possible transitions to its Upstream(S,G) state machine.

Current Assert Winner's NeighborLiveness Timer Expires

The current Assert winner's NeighborLiveness Timer (NLT(N,I)) has expired. The Assert state machine MUST transition to the NoInfo (NI) state, delete the Assert Winner's address and metric, and evaluate any possible transitions to its Upstream(S,G) state machine..

Receive Prune(S,G), Join(S,G) or Graft(S,G)

A Prune(S,G), Join(S,G) or Graft(S,G) message was received on interface I with its upstream neighbor address set to the router's address on I. The router MUST send an Assert(S,G) on the receiving interface I to initiate an Assert negotiation. The Assert state machine remains in the Assert Loser(L) state.

6.6.5. Rationale for Assert Rules

The following is a summary of the rules for generating and processing Assert messages. It is not intended to be definitive (the state machines and pseudocode provide the definitive behavior). Instead it provides some rationale for the behavior.

1. The Assert winner for (S,G) must act as the local forwarder for (S,G) on behalf all downstream members.

2. PIM messages are directed towards to the RPF' neighbor and not to the regular RPF neighbor.

3. An Assert loser that receives a Prune(S,G), Join(S,G) or Graft(S,G) directed to it initiates a new Assert negotiation so the downstream router can correct its RPF'(S).

4. An assert winner for (S,G) sends a canceling assert when it is about to stop forwarding on an (S,G) entry. Example : if a router is being taken down, then a canceling assert is sent.

5.

6.7. PIM Packet Formats

All PIM-DM packets use the same format as PIM-SM packets. All PIM control messages have IP protocol number 103. All PIM-DM messages MUST be sent with a TTL of 1. All PIM-DM messages except Graft and Graft Ack messages MUST be sent to the ALL-PIM-ROUTERS group (224.0.0.13). Graft messages SHOULD be unicast to the RPF'(S). Graft Ack messages MUST be unicast to the sender of the Graft.

6.7.1. PIM Header

All PIM control messages have the following header:

```
 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            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

PIM Ver
PIM version number is 2.

Type
Types for specific PIM messages. Available types are:
0 = Hello
1 = Register (PIM-SM only)
2 = Register Stop (PIM-SM only)
3 = Join/Prune
4 = Bootstrap (PIM-SM only)
5 = Assert
6 = Graft
7 = Graft Ack
8 = Candidate RP Advertisement (PIM-SM only)
9 = State Refresh

Reserved

Set to zero on transmission. Ignored upon receipt.

Checksum

The checksum is standard IP checksum, i.e. the 16 bit one's complement of the one's complement sum of the entire PIM message. For computing checksum, the checksum field is zeroed.

6.7.2. Encoded Unicast Address

An Encoded Unicast Address has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Addr Family  | Encoding Type |     Unicast Address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...
```

Addr Family

The PIM Address Family of the 'Unicast Address' field of this address.

Values of 0-127 are as assigned by the IANA for Internet Address Families in [6]. Values 128-250 are reserved to be assigned by the IANA for PIM specific Address Families. Values 251-255 are designated for private use. As there is no assignment authority for this space, collisions should be expected.

Encoding Type

The type of encoding used with a specific Address Family. The value '0' is reserved for this field, and represents the native encoding of the Address Family

Unicast Address

The unicast address as represented by the given Address Family and Encoding Type.

6.7.3. Encoded Group Address

An Encoded Group address has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Addr Family  | Encoding Type |   Reserved    |  Mask Len     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Group Multicast Address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+...
```

Addr Family
As described above.

Encoding Type

As described above.

Reserved

Transmitted as zero. Ignored upon receipt.

Mask Len

The mask length field is 8 bits. The value is the number of contiguous on bits left justified used as a mask, which combined with the address, describes a range of addresses. It is less than or equal to the address length in bits for the given Address Family and Encoding Type. If the message is sent for a single address then the mask length MUST equal the address length. PIM-DM routers MUST only send for a single address.

Group Multicast Address

The address of the multicast group.

6.7.4. Encoded Source Address

An Encoded Source address has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Addr Family  | Encoding Type |  Rsrvd  |S|W|R|  Mask Len     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Source Address
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-...
```

Addr Family

As described above.

Encoding Type

As described above.

Rsrvd

Reserved. Transmitted as zero. Ignored upon receipt.

S

The Sparse Bit. Set to 0 for PIM-DM. Ignored upon receipt.

W

The Wild Card Bit. Set to 0 for PIM-DM. Ignored upon receipt.

R

The Rendezvous Point Tree bit. Set to 0 for PIM-DM. Ignored upon receipt.

Mask Len

As described above. PIM-DM routers MUST only send for a single source address.

Source Address

The source address.

6.7.5. Hello Message Format

The PIM Hello message has the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|PIM Ver| Type  |   Reserved    |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Option Type          |         Option Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Option Value                          |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                               .                               |
|                               .                               |
|                               .                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Option Type          |         Option Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Option Value                          |
|                              ...                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

PIM Ver, Type, Reserved, Checksum
Described above.

Option Type
The type of option given in the Option Value field. Available types are:

0	Reserved
1	Hello Hold Time
2	LAN Prune Delay
3-16	Reserved
17	To be assigned by IANA
18	Deprecated and SHOULD NOT be used
19	DR Priority (PIM-SM Only)
20	Generation ID
21	State Refresh Capable
22-65000	To be assigned by IANA
65001-65535	Reserved for Private Use [7]

Unknown options SHOULD be ignored.

6.7.5.1. Hello Hold Time Option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 1           |           Length = 2          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Hold Time          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Hold Time is the number of seconds a receiver MUST keep the neighbor reachable. If the Hold Time is set to '0xffff', the receiver of this message never times out the neighbor. This may be used with dial-on-demand links, to avoid keeping the link up with periodic Hello messages. Furthermore, if the Holdtime is set to '0', the information is timed out immediately. The Hello Hold Time option MUST be used by PIM-DM routers.

6.7.5.2. LAN Prune Delay Option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Type = 2           |           Length = 4          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|T|       LAN Prune Delay       |       Override Interval       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The LAN_Prune_Delay option is used to tune the prune propagation delay on multi-access LANs. The T bit is used by PIM-SM and SHOULD be set to 0 by PIM-DM routers and ignored upon receipt. The LAN Delay and Override Interval fields are time intervals in units of milliseconds and are used to tune the value of the J/P Override Interval and its derived timer values. Section 6.3.5 describes how these values affect the behavior of a router. The LAN Prune Delay SHOULD be used by PIM-DM routers.

6.7.5.3. Generation ID Option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 20           |           Length = 4          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Generation ID                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Generation ID is a random value for the interface on which the Hello message is sent. The Generation ID is regenerated whenever PIM forwarding is started or restarted on the interface. The Generation ID option MAY be used by PIM-DM routers.

6.7.5.4. State Refresh Capable Option

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 21           |           Length = 4          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Version = 1  |   Interval    |            Reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Interval field is the router’s configured State Refresh Interval in seconds. The Reserved field is set to zero and ignored upon reception. The State Refresh Capable option MUST be used by State Refresh capable PIM-DM routers.

6.7.6. Join/Prune Message Format

PIM Join/Prune messages have the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|PIM Ver| Type  |   Reserved    |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Upstream Neighbor Address (Encoded Unicast Format)     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Reserved    |  Num Groups   |          Hold Time            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         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)       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

PIM Ver, Type, Reserved, Checksum
Described above.

Upstream Neighbor Address
The address of the upstream neighbor. The format for this address is given in the Encoded Unicast address in section 6.7.2. PIM-DM routers MUST set this field to the RPF next hop.

Reserved
Transmitted as zero. Ignored upon receipt.

Hold Time
The number of seconds a receiving PIM-DM router MUST keep a Prune state alive, unless removed by a Join or Graft message. If the Hold Time is '0xffff', the receiver MUST NOT remove the Prune state unless a corresponding Join or Graft message is received. The Hold Time is ignored in Join messages.

Number of Groups
Number of multicast group sets contained in the message.

Multicast Group Address
The multicast group address in the Encoded Multicast address format given in section 6.7.3.

Number of Joined Sources
Number of Join source addresses listed for a given group.

Number of Pruned Sources
Number of Prune source addresses listed for a given group.

Join Source Address 1..n
This list contains the sources from which the sending router wishes to continue to receive multicast messages for the given group on this interface. The addresses use the Encoded Source address format given in section 6.7.4.

Prune Source Address 1..n
This list contains the sources from which the sending router does not wish to receive multicast messages for the given group on this interface. The addresses use the Encoded Source address format given in section 6.7.4.

6.7.7. Assert Message Format

PIM Assert Messages have the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|PIM Ver| Type  |   Reserved    |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Multicast Group Address (Encoded Group Format)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Source Address (Encoded Unicast Format)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|                     Metric Preference                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Metric                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

PIM Ver, Type, Reserved, Checksum
Described above.

Multicast Group Address
The multicast group address in the Encoded Multicast address format given in section 6.7.3.

Source Address
The source address in the Encoded Source address format given in section 6.7.4.

R
The Rendezvous Point Tree bit. Set to 0 for PIM-DM. Ignored upon receipt.

Metric Preference
The preference value assigned to the unicast routing protocol that provided the route to the source.

Metric
The cost metric of the unicast route to the source. The metric is in units applicable to the unicast routing protocol used.

6.7.8. Graft Message Format

PIM Graft messages use the same format as Join/Prune messages except the Type field is set to 6. The source address MUST be in the Join section of the message. The Hold Time field SHOULD be zero and SHOULD be ignored when a Graft is received.

6.7.9. Graft Ack Message Format

PIM Graft Ack messages are identical in format to the received Graft message except the Type field is set to 7. The Upstream Neighbor Address field SHOULD be set to the sender of the Graft message and SHOULD be ignored upon receipt.

6.6.10. State Refresh Message Format

PIM State Refresh Messages have the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|PIM Ver| Type  |   Reserved    |           Checksum            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Multicast Group Address (Encoded Group Format)        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Source Address (Encoded Unicast Format)           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Originator Address (Encoded Unicast Format)         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|R|                     Metric Preference                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             Metric                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Masklen    |    TTL        |P|N|O|Reserved |   Interval    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

PIM Ver, Type, Reserved, Checksum
Described above.

Multicast Group Address

The multicast group address in the Encoded Multicast address format given in section 6.7.3.

Source Address
The address of the data source in the Encoded Source address format given in section 6.7.4.

Originator Address
The address of the first hop router in the Encoded Source address format given in section 6.7.4.

R
The Rendezvous Point Tree bit. Set to 0 for PIM-DM. Ignored upon receipt.

Metric Preference
The preference value assigned to the unicast routing protocol that provided the route to the source.

Metric
The cost metric of the unicast route to the source. The metric is in units applicable to the unicast routing protocol used.

Masklen
The length of the address mask of the unicast route to the source.

TTL
Time To Live of the State Refresh message. Decremented each time the message is forwarded. Note that this is different from the IP Header TTL, which is always set to 1.

P
Prune indicator flag. This MUST be set to 1 if the State Refresh is to be sent on a Pruned interface. Otherwise, it MUST be set to 0.

N
Prune Now flag. This SHOULD be set to 1 by the State Refresh originator on every third State Refresh message and SHOULD be ignored upon receipt. This is for compatibility with earlier versions of state refresh.

O
Assert Override flag. This SHOULD be set to 1 by upstream routers on a LAN if a State Refresh has not been heard from the assert winner over the period of three times RefreshInterval(S,G) and SHOULD be ignored upon receipt. This is for compatibility with earlier versions of state refresh.

Reserved
Set to zero and ignored upon receipt.

Interval
Set by the originating router to the interval (in seconds) between consecutive State Refresh messages for this (S,G) pair.

6.8. PIM-DM Timers

PIM-DM maintains the following timers. All timers are countdown timers – they are set to a value and count down to zero, at which point they typically trigger an action. Of course they can just as easily be implemented as count-up timers, where the absolute expiry time is stored and compared against a real-time clock, but the language in this specification assumes that they count downwards towards zero.

Global Timers

Hello Timer: HT

Per interface (I):
Propagation Delay: PD(I)
Override Interval: OI(I)

Per neighbor (N):

Neighbor Liveness Timer: NLT(N,I)

Per (S,G) Pair:

(S,G) Assert Timer: AT(S,G,I)
(S,G) Prune Timer: PT(S,G,I)
(S,G) PrunePending Timer: PPT(S,G,I)

Per (S,G) Pair:
(S,G) Graft Retry Timer: GT(S,G)
(S,G) Upstream Override Timer: OT(S,G)
(S,G) Prune Limit Timer: PLT(S,G)
(S,G) Source Active Timer: SAT(S,G)
(S,G) State Refresh Timer: SRT(S,G)

6.8.1. Timer Values

When timer values are started or restarted, they are set to default values. This section summarizes those default values.

Timer Name: Hello Timer (HT)

Value Name	Value	Explanation
Hello_Period	30 sec	Periodic interval for hello messages
Triggered_Hello_Delay	5 sec	Random interval for initial Hello message on bootup or triggered Hello message to a rebooting neighbor

Hello message are sent on every active interface once every Hello_Period seconds. At system power-up, the timer is initialized to rand(0,Triggered_Hello_Delay) to prevent synchronization. When a new or rebooting neighbor is detected, a responding Hello is sent within rand(0,Triggered_Hello_Delay). Hello_Period corresponds to the PIM MIB object pimInterfaceHelloInterval.

Timer Name: Propagation Delay (PD(I))

Value Name	Value	Explanation
LAN_delay_default	0.5 sec	Expected propagation over the local link.

If all routers on a LAN are using the LAN Prune Delay option, PD(I) will be set to the largest LAN Delay on the LAN. Otherwise, PD(I) will be set to LAN_delay_default.

Timer Name: Override Interval (OI(I))

Value Name	Value	Explanation
t_override_default	2.5 sec	Default delay interval over which to randomize when scheduling a Join/ Prune Override message.

If all routers on a LAN are using the LAN Prune Delay option, OI(I) will be set to the largest Override Interval on the LAN. Otherwise, OI(I) will be set to t_override_default.

Timer Name: Neighbor Liveness Timer (NLT(N,I))

Value Name	Value	Explanation
Hello Holdtime	From message	Hold Time from Hello Message

Timer Name: PrunePending Timer (PPT(S,G,I))

Value Name	Value	Explanation
J/P_Override_Interval	OI(I) + PD(I)	Short time after a Prune to allow other routers to on the LAN to send a Join

Timer Name: Prune Timer (PT(S,G,I))

Value Name	Value	Explanation
Prune Holdtime	From message	Hold Time read from Prune Message

Prune Holdtime corresponds to PIM MIB object pimInterfaceJoinPruneInterval

Timer Name: Assert Timer (AT(S,G,I))

+-------------------------+---------+-------------------------------+

Value Name	Value	Explanation
Assert Override Interval	3 sec	Short interval before an assert times out where the assert winner resends an assert message
Assert Time	180 sec	Period after last assert before assert state is timed out

Note that for historical reasons, the Assert message lacks a Holdtime field. Thus changing the Assert Time from the default value is not recommended. If all members of a LAN are state refresh enabled, the Assert Time will be three times RefreshInterval(S,G) and Assert Override will not be needed.

Timer Name: Graft Retry Timer (GRT(S,G))

Value Name	Value	Explanation
Graft_Retry_Period	3 sec	In the absence of receipt of a GraftAck message, the time before retransmission of a Graft message

Timer Name: Upstream Override Timer (OT(S,G))

Value Name	Value	Explanation
t_override	rand(0, OI(I))	Randomized delay to prevent response implosion when sending a join message to override someone else's prune

Timer Name: Prune Limit Timer (PLT(S,G))

Value Name	Value	Explanation
t_limit	Equal to the Prune Holdtime sent	Used to prevent Prune storms on a LAN

Timer Name: Source Activity Timer (SAT(S,G))

Value Name	Value	Explanation
SourceLifetime	Default: 210 secs	Period of time after receiving a multicast message a directly attached router will continue to send State Refresh messages

Timer Name: State Refresh Timer (SRT(S,G))

Value Name	Value	Explanation
RefreshInterval	Default: 60 secs	Interval between successive state refresh messages

7. Protocol Interaction Considerations

PIM-DM is designed to be independent of underlying unicast routing protocols and will interact only to the extent needed to perform RPF checks. It is generally assumed that multicast area and autonomous system boundaries will correspond to the same boundaries for unicast routing, though a deployment which does not follow this assumption is not precluded by this specification.

In general, PIM-DM interactions with other multicast routing protocols should be in compliance with RFC 2715 [13]. Other specific interactions are noted below.

7.1. PIM-SM Interactions

PIM-DM is not intended to interact directly with PIM-SM, even though they share a common packet format. It is particularly important to note that a router cannot differentiate between a PIM-DM neighbor and a PIM-SM neighbor based on Hello messages.

In the event that a PIM-DM router becomes a neighbor of a PIM-SM router they will effectively form a simplex link with the PIM-DM router sending all multicast messages to the PIM-SM router while the PIM-SM router sends no multicast messages to the PIM-DM router.

The common packet format permits a hybrid PIM-SM/DM implementation that would use PIM-SM when a rendezvous point is known and PIM-DM when one is not. Such an implementation is outside the scope of this document.

7.2. IGMP Interactions

PIM-DM will forward received multicast data packets to neighboring host group members in all cases except when the PIM-DM router is in an Assert Loser state on that interface. Note that a PIM Prune message is not permitted to prevent the delivery of messages to a network with group members.

A PIM-DM Router MAY use the DR Priority option described in [3] to elect an IGMP v1 querier.

7.3. Source Specific Multicast (SSM) Interactions

PIM-DM makes no special considerations for SSM [11]. All Prunes and Grafts within the protocol are for a specific source, so no additional checks need be made.

7.4. Multicast Group Scope Boundary Interactions

While multicast group scope boundaries are generally identical to routing area boundaries, it is conceivable that a routing area might be partitioned for a particular multicast group. PIM-DM routers MUST NOT send any messages concerning a particular group across that group’s scope boundary.

7.5. IANA Considerations

7.5.1. PIM Address Family

The PIM Address Family field was chosen to be 8 bits as a tradeoff between packet format and use of the IANA assigned numbers. When the PIM packet format was designed, only 15 values were assigned for Address Families and large numbers of new Address Families were not envisioned, 8 bits seemed large enough. However, the IANA assigns Address Families in a 16 bit value. Therefore, the PIM Address Family is allocated as follows:

Values 0 through 127 are designated to have the same meaning as IANA assigned Address Family Numbers [6].

Values 128 through 250 are designated to be assigned by the IANA based upon IESG approval as defined in [7].

Values 251 through 255 are designated for Private Use, as defined in [7].

7.6. PIM Hello Options

Values 17 through 65000 are to be assigned by the IANA. Since the space is large, they may be assigned as First Come First Served as defined in [7]. Such assignments are valid for one year, and may be renewed. Permanent assignments require a specification as defined in [7].

8. Security Considerations

All PIM control messages MAY use IPsec [8] to address security concerns. The authentication methods are addressed in a companion document [9]. Keys may be distributed as described in [10]. In any case, PIM router SHOULD NOT accept and process PIM messages from neighbors unless a valid Hello message has been received from that neighbor.
[*We need to verify that [9] and [10] are not PIM-SM specific.*]

We should note that PIM-DM has no rendezvous point, and therefore no single point of failure that may be vulnerable. However, since PIM-DM assumes that multicast messages are desired throughout the network, it may be particularly vulnerable to denial of service attacks.

It is further worth noting that because PIM-DM uses unicast routes provided by an unknown routing protocol, it may suffer collateral effects if the unicast routing protocol is attacked.

9. Authors' Addresses

Andrew Adams
NextHop Technologies
825 Victors Way, Suite 100
Ann Arbor, MI 48108-2738
ala@nexthop.com

Jonathan Nicholas
ITT Aerospace/Communications Division
100 Kingsland Rd
Clifton, NJ 07014
jonathan.nicholas@itt.com

William Siadak
NextHop Technologies
825 Victors Way, Suite 100
Ann Arbor, MI 48108-2738
wfs@nexthop.com

10. Acknowledgments

The major features of PIM-DM were originally designed by Stephen Deering, Deborah Estrin, Dino Farinacci, Van Jacobson, Ahmed Helmy, David Meyer, and Liming Wei. Additional features for state refresh were designed by Dino Farinacci, Isidor Kouvelas and Kurt Windisch. This revision was undertaken to incorporate some of the lessons learned during the evolution of the PIM-SM specification and early deployments of PIM-DM.

Thanks the PIM Working Group for their comments.

12. References

[1] S.E. Deering, "Multicast Routing in a Datagram Internetwork", Ph.D. Thesis, Electrical Engineering Dept., Stanford University, December 1991.

[2] D. Waitzman, B.Partridge, S.Deering, "Distance Vector Multicast Routing Protocol", November 1988, RFC 1075

[3] W. Fenner, M. Handley, H.Holbrook, I. Kouvelas, "Protocol Independent Multicast - Sparse Mode (PIM-SM)", draft-ietf-pim-sm-v2-new-02.txt, work in progress.

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

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

[6] IANA, "Address Family Numbers", linked from http://www.iana.org/numbers.html.

[7] T. Narten, H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434.

[8] S. Kent, R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401.

[9] L. Wei, "Authenticating PIM Version 2 Messages", draft-ietf-pim-v2-auth-01.txt, work in progress.

[10] T. Hardjono, B. Cain, "Simple Key Management Protocol for PIM", draft-ietf-pim-simplekmp-01.txt, work in progress.

[11] H.Holbrook, B. Cain, "Source Specific Multicast for IP", draft-holbrook-ssm-00.txt, work in progress.

[12] B. Cain, S. Deering, B. Fenner, I. Kouvelas, A. Thyagarajan, “Internet Group Management Protocol, Version 3”, draft-ietf-idmr-igmp-v3-07.txt, work in progress.

[13] D. Thaler, “Interoperability Rules for Multicast Routing Protocols”, October 1999, RFC 2715

[14] K.McCloghrie, D.Farinacci, D.Thaler, B.Fenner, “Protocol Independent Multicast MIB for IPv4”, October 2000, RFC 2934