HPIM-DM state machines

Pedro Francisco Carmelo de Oliveira
pedro.francisco.oliveira@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

March 14, 2020

This document defines all specified state machines:

- **Section 1** defines all state that must be stored at routers;
- **Section 2** defines all events that the protocol must react to;
- **Section 3** describes how message sequencing is performed;
- **Section 4** defines the transmission of Hello messages;
- **Section 5** defines the state machines related with the formation and maintenance of neighborhood relationships (synchronization process and reception of Hello messages);
- **Section 6** defines the maintenance of upstream and interest state of neighbors through the reception of control messages, along with the transmission of ACKs and maintenance of SN information;
- **Section 7** defines the state of trees;
- **Section 8** defines the Assert state of non-root interfaces;
- **Section 9** defines the Downstream Interest state of non-root interfaces;
- **Section 10** defines the Forwarding state of non-root interfaces;
- **Section 11** defines the interest state of the own router;
- **Section 12** defines when an interface must transmit control messages;
- **Section 13** defines reliability of transmitted control messages;
- **Section 14** defines when routers can remove state about neighbors/trees;
- **Section 15** defines an optimization to remove sequence information (CheckpointSN);
- **Section 16** defines the default values of timers.
- **Section 17** defines the format of HPIM-DM control messages.

1 Stored state

The per interface information is:

- **BootTime** – Timestamp obtained when the interface booted up or when the SN overflowed;
- **InterfaceSN** – Last transmitted SN;
- **HelloTimer** – Timer used to control transmission of Hello messages;
- **NeighborList** - List/Dictionary with information regarding all known neighbors of this interface.

The per neighbor information is:

- **State** – Synchronization state;
- **BootTime** – Timestamp obtained when the neighbor's interface booted up. Used to detect reboots/sequence number overflow and to filter messages transmitted prior to the synchronization;
- **MySnapshotSN** – SN when the own router created a snapshot, used to synchronize with this neighbor;
- **NeighborSnapshotSN** – SN of the neighbor router when it took its snapshot;
- **CheckpointSN** – SN of the neighbor router, used to save resources regarding stored sequence numbers;
- **CurrentSyncSN** – SN used for the fragmentation of Sync messages;
- **SyncRetransmissionTimer** – Timer used to regulate the retransmission of Sync messages, while the synchronization is occurring with the neighbor;
- **NeighborLivenessTimer** – Timer used for determining when the neighbor is declared to have failed;
- **Snapshot** – Snapshot of all trees in which the router is considered as UPSTREAM to be transmitted to the neighbor during the synchronization process (only required during an ongoing synchronization).

The per neighbor and per (S,G) tree information is:

- **UpstreamState** – Whether the neighbor is considered to be UPSTREAM or NOT UPSTREAM for (S,G) tree. If it is UPSTREAM, the RPC is also required to be stored;
- **InterestState** – Whether the neighbor is considered to be INTERESTED or NOT INTERESTED for (S,G) tree;
- **LastSN** – Last received SN in control messages from the neighbor regarding (S,G) tree. This SN is only required to be stored if it is greater than the CheckpointSN stored for the neighbor.

The per router information is:

- **Trees** - All known trees that are currently being maintained (that are in state different than INACTIVE).

The per tree information is:

- **State** – State of the tree (ACTIVE/UNSURE/INACTIVE), depending on the existence of Upstream neighbors connected to an interface and also whether the router is considered to be or not an originator;
- **RouterInterest** - Interest of router in receiving data packets;
- **MyRPC** – RPC of the own router to the source S (via unicast routing table).

The per interface and per tree information is:

- **Role** – Whether the interface is root or non-root;
- **MessagesPendingAcknowledgement** – All messages that are currently being reliably transmitted and that are pending the acknowledgment from neighbors;
- **ReliableTransmissionTimer** – Timer used to regulate the retransmission of control messages pending acknowledgment.

The per root interface and per tree information of non-originators' routers is:

- **BestUpstreamNeighbor** – Neighbor connected to the root interface that considers itself as UPSTREAM and offers the best RPC to source S. This router will be responsible for forwarding data packets to the link that connects this root interface. Interest and NoInterest messages should be destined to this router;

The per root interface and per tree information of originators' routers is:

- **SourceActiveTimer (SAT)** – Timer used to determine when the tree/source is no longer declared to be active.

The per non-root interface is:

- **BestUpstreamNeighbor** – Neighbor connected to a non-root interface that considers itself as UPSTREAM and offers the best RPC to the source S. This information will be used to determine if this interface should be considered the Assert Winner or Assert Loser of the link and also to transmit NoInterest messages in case the router considers the tree to be in UNSURE state.
- **AssertState** - Whether the interface is AW or AL;
- **DownstreamInterestState** - Whether the interface is DOWNSTREAM INTERESTED or NOT DOWNSTREAM INTERESTED. This is obtained from the interest state of all neighbor and also the MLD/IGMP state machine;
- **MLD/IGMP State** - State of MLD/IGMP regarding this tree or group. Used to determine the interest of directly connected hosts;
- **ForwardingState** - Whether the interface is FORWARDING or PRUNED.

2 Definition of Events

Our protocol reacts to the following events:

- Reception or absence of data messages;
- Reception of control messages;
- New neighbor;
- Failure of neighbor;
- Change of RPC;
- Change of interfaces' roles;
- New interface;
- Interface removal;
- IGMP/MLD state transitions.

Any of the above mentioned events may cause the discovery of new trees, changes in existing trees or the complete removal of a tree. When any of these events happens, a bunch of verifications must occur through all state machines. Figure 1 defines the order by which these verification must occur. This order must be respected since state machines below this hierarchy are dependent on the state of state machines above them. For example, the router interest is dependent on the Forwarding state of non-root interfaces.

3 Message sequencing

Each interface requires to store its BootTime.

Additionally, each interface must also store its last transmitted SN. This stored information will be called InterfaceSN. Following control messages and synchronization processes will use this information (increment the InterfaceSN).

The CheckpointSN is calculated per interface having in consideration all messages that are currently being transmitted and all message that have already been acknowledged. This information does not require to be stored and can be calculated on-demand (when it is required to be transmitted).

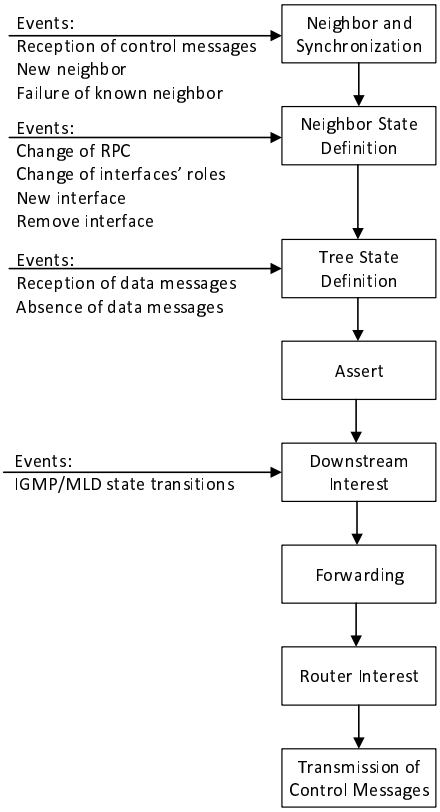

Figure 1: Events and corresponding order by which state machines must be verified.

4 Hello protocol

This mechanism is used to discover new routers and also to detect failures of already known neighbors. By not hearing Hello messages from a given neighbor, we suspect that it has failed.

Table 1 describes when Hello messages should be transmitted. This state machine uses one timer to regulate the transmission of those messages, called Hello Timer (HT).

The Hello messages must include the BootTime in order for neighbors to detect reboots/sequence number overflows. These messages do not include the other sequence number (SN), since they are not transmitting state. These messages must include the Hello Hold Time option to advertise to all neighbors their transmission periodicity of Hello messages (to detect failures). The Hello Hold Time must be 3.5 times the periodicity of Hello messages.

	Event	Action
1	Start routing process in interface I	Send Hello message Set HT
2	HT expires	Send Hello message Set HT
3	Interface I no longer participates in the routing process	Send Hello message informing that I no longer participates in routing process Clear HT

Table 1: Transmission of Hello messages.

In event 1 of Table 1, when a new interface is enabled in the multicast routing process, this interface requires to transmit an Hello message, in order to establish neighborhood relationships. Besides transmitting the message, the interface sets the HT, used to regulate the periodic transmission of those types of messages.

In event 2 of Table 1, when the HT expires, a new Hello message must be transmitted. The HT is

reset, in order to schedule the next Hello message transmission.

In event 3 of Table 1, when an interface no longer participates in the multicast routing process, an Hello message should be transmitted in order to notify its neighbors. This is just to accelerate the removal of the neighbor, instead of relying on the absence of reception of Hello messages. Neighbors can be informed that a neighbor is being removed, by transmitting an Hello message with zero as lifetime (Hello Hold Time), like in PIM-DM. Since this interface will no longer transmit Hello messages, the HT is canceled.

Regarding the detection of new neighbors, this is described in the next subsection, by the Synchronization mechanism.

5 Neighbors and Synchronization mechanism

Each neighbor can be in one of four states:

- UNKNOWN - unknown neighbor;
- MASTER - neighbor is in an ongoing synchronization process and is the Master of this process;
- SLAVE - neighbor is in an ongoing synchronization process and is the Slave of this process;
- SYNCED - neighbor has successfully synchronized.

All neighbors are initially in UNKNOWN state, until they are discovered through the exchange of control messages. When a new synchronization process is triggered the neighbor transitions to a MASTER or SLAVE state, according to the one that initiates this process or according to the IP address of both routers. After the synchronization finishes successfully the neighbor is placed in SYNCED state and will remain in this state as long as it is alive. By considering a neighbor to have failed, it transitions to UNKNOWN state, regardless of which state it is at.

A router must store all types of sequence numbers, regarding itself and the neighbor router:

- neighbor's BootTime
- MySnapshotSN
- NeighborSnapshotSN
- CurrentSyncSN

Additionally, there are two specific timers used for each neighbor:

- NeighborLivenessTimer (NLT) - Used to detect failures of a known neighbor;
- SyncRetransmissionTimer - Used to control the retransmission of previously sent Sync messages, in case there is no proper response from the neighbor.

The snapshot that must be taken by a router when a synchronization process is triggered by its interface consists of all trees in which:

- the interface has a non-root role, AND
- interface is not directly connected to the source;

Each synchronization process (even concurrently) has a unique snapshot. The snapshot (for a given synchronization process) must only be stored during this process; After the synchronization between two routers finishes successfully, this stored information can be removed.

Listing 1, describes in pseudocode how to process received Sync and Hello messages from a neighbor router and when its state must change.

```
New neighbor detected via non-Sync message or                                    1
NeighborSnapshotSN changes via Sync message or BootTime                          2
changes during or after synchronization (via Sync or non-Sync message):         3
   -> SLAVE                                                                      4
   Clear stored info from neighbor                                               5
   MySnapshotSN = InterfaceSN++                                                  6
   Take snapshot                                                                 7
   CurrentSyncSN = 0                                                             8
   Set SyncRetransmissionTimer                                                   9
   Set NLT                                                                       10
   Send Sync                                                                     11
                                                                                 12
Receive Sync message:                                                            13
   If neighbor is SLAVE and Interface.BootTime == Message.NeighborBootTime and   14
         NeighborBootTime == Message.MyBootTime and                              15
         Message.SyncSN == CurrentSyncSN:                                        16
      If Message.SyncSN == 0 and Message.MasterBit:                              17
         If MyIP < NeighborIP:                                                   18
            -> MASTER                                                            19
            Send Sync                                                            20
         Else:                                                                   21
            Resend last transmitted message                                      22
      Else If not Message.MasterBit and                                          23
            NeighborSnapshotSN == Message.MySnapshotSN and                       24
            MySnapshotSN == Message.NeighborSnapshotSN:                          25
         If Message.SyncSN > 0 and not Message.MoreBit and                       26
               ''I dont need to transmit more messages'':                        27
            -> SYNCED                                                            28
            Set NLT                                                              29
            Clear SyncRetransmissionTimer                                        30
         Else:                                                                   31
            CurrentSyncSN++                                                      32
            Set NLT                                                              33
            Set SyncRetransmissionTimer                                          34
            Send Sync                                                            35
                                                                                 36
   If neighbor is MASTER and Interface.BootTime == Message.NeighborBootTime and  37
         NeighborBootTime == Message.MyBootTime and                              38
         NeighborSnapshotSN == Message.MySnapshotSN and                          39
         Message.SyncSN == CurrentSyncSN:                                        40
      If Message.MasterBit and (Message.SyncSN > 0 and                           41
            MySnapshotSN == Message.NeighborSnapshotSN                           42
            or Message.SyncSN == 0):                                             43
         Send Sync                                                               44
         If Message.SyncSN > 0 and not Message.MoreBit                           45
               and ''I dont need to transmit more messages'':                    46
            -> SYNCED                                                            47
            Clear SyncRetransmissionTimer                                        48
         Else:                                                                   49
            CurrentSyncSN++                                                      50
                                                                                 51
   If neighbor is SYNCED:                                                        52
      If Message.MasterBit and CurrentSyncSN == Message.SyncSN and               53
            Interface.BootTime == Message.NeighborBootTime and                   54
            NeighborBootTime == Message.MyBootTime and                           55
            MySnapshotSN == Message.NeighborSnapshotSN and                       56
            NeighborSnapshotSN == Message.MySnapshotSN:                          57
```

```
        Resend last transmitted message                                          58
                                                                                 59
    If neighbor is UNKNOWN and InterfaceBootTime == Message.NeighborBootTime:    60
        If Message.SyncSN == 0 and Message.MasterBit:                            61
            -> MASTER                                                            62
            MySnapshotSN = InterfaceSN++                                         63
            Take snapshot                                                        64
            CurrentSyncSN = 0                                                    65
            Send Sync                                                            66
            Set NLT                                                              67
            Set SyncRetransmissionTimer                                          68
        Else:                                                                    69
            -> SLAVE                                                             70
            MySnapshotSN = InterfaceSN++                                         71
            Take snapshot                                                        72
            CurrentSyncSN = 0                                                    73
            Send Sync                                                            74
            Set NLT                                                              75
            Set SyncRetransmissionTimer                                          76
                                                                                 77
SyncRetransmissionTimer expires:                                                 78
    Resend last transmitted message                                              79
    Set SyncRetransmissionTimer                                                  80
                                                                                 81
NLT expires:                                                                     82
    -> UNKNOWN                                                                   83
    Clear SyncRetransmissionTimer                                                84
    Clear stored info from neighbor                                              85
                                                                                 86
Receive Hello message:                                                           87
    If neighbor is SYNCED:                                                       88
        Set NLT                                                                  89
```

Listing 1: Pseudocode used in the synchronization process.

1) A new synchronization process must start, in which the router declares itself to be Master, when:

- New neighbor is detected via non-Sync message - by discovering a new neighbor that is not triggering the synchronization process, the interface takes the initiative, by declaring itself to be the Master of the synchronization process.
- Known neighbor sends a Sync message with same BootTime but a greater SnapshotSN - by hearing a Sync message, in which the known neighbor declares a greater SnapshotSN, this means that it is triggering a new synchronization process.
- Known neighbor sends any type of control message with a greater BootTime - by hearing any type of control message, Sync or non-Sync message, with a greater BootTime, a new synchronization process must be started. Since there is no way to distinguish between router resets and sequence number overflows, a greater BootTime triggers always a new synchronization.

In the last two cases, if a router hears a Sync message from a known neighbor, that is triggering a new synchronization process, the router must declare itself to be the Master of this process. Both routers will initially declare to be Masters of this process, being the IP address used to break the "tie".

In any of the three cases described, the router must take a snapshot of its trees, obtain a SnapshotSN (which will be called MySnapshotSN), consider the neighbor router to be in SLAVE state and transmit a Sync message to it. The first Sync message must have information regarding the neighbor's BootTime, the own router's BootTime and MySnapshotSN. The NeighborSnapshotSN is not known, thus set to 0. The NeighborLivenessTimer and SyncRetransmissionTimer are set, for detecting neighbor failures during the synchronization and to control the retransmission of Sync messages, respectively.

2) When a router hears a Sync message that has correct information regarding the own router's BootTime, the neighbor's BootTime, the SyncSN (according to the CurrentSyncSN) and the NeighborSnapshotSN (except for the first transmitted message):

- From a neighbor that is considered to be in SLAVE state:
  - If that neighbor transmits a Sync message declaring to be the Master of the synchronization process (and if it is the first message - SyncSN=0), this means that both neighbors consider themselves as Masters of this process. In order to have progress in this process, the IP address of both neighbors is used, being the neighbor with the greatest IP address the Master and the other the Slave. If the router that receives this message has the lowest IP address, the neighbor is placed in MASTER state, and a Sync message is transmitted (in which it does not declare to be Master - Master flag cleared).
  - If the neighbor transmits a message in which it does not declare to be the Master of that process, it means that the synchronization is making progress. If no more messages are required to be transmitted (SyncSN>0, received More flag cleared and the last transmitted message, by the own router, had the More flag cleared), the neighbor is placed in SYNCED state, otherwise a Sync message must be transmitted.

    The SyncSN is required to be greater than 0 in order to terminate the synchronization process because both routers are required to reliably transmit all sequence numbers, which is only guaranteed if the SyncSN is greater than 0.
- From a neighbor that is considered to be in MASTER state:

  In this case, the received Sync message must have the Master flag set. Also, the MySnapshotSN (received message's NeighborSnapshotSN) must be correct (if the received SyncSN is greater than 0).

  If it is guaranteed that the message corresponds to the current synchronization process and all sequence numbers are correct, a newer Sync message is transmitted, in response to the Master's message (with same SyncSN).
  - If both routers transmitted messages with the More flag cleared and the SyncSN is greater than 0, this means that the synchronization process has finished and the neighbor is placed in SYNCED state.
  - Otherwise, the router will wait for the reception of more Sync messages from it.
- If the neighbor is in SYNCED state and receives a Sync message having all sequence numbers correct:

  The router should resend the last transmitted Sync message if the neighbor declares itself to be Master (Master flag set). This retransmission is required because there is the possibility that the last transmitted Sync message, from the Slave, was lost.
- If the neighbor is in UNKNOWN state and receives a Sync message (with the correct BootTime):

  If the neighbor declares itself as Master (Master flag set) and is its first transmitted Sync message (SyncSN=0), then the router declares itself as the Slave of the synchronization process, meaning that the neighbor is placed in MASTER state and a newer snapshot is taken with its corresponding sequence number (MySnapshotSN). A Sync message is transmitted to the neighbor router declaring itself as Slave (Master flag cleared). Information about the SNs of the neighbor router (BootTime and neighbor's SnapshotSN), present in the received Sync message, must be stored.

  If the neighbor does not declare itself as Master (Master flag cleared) or is not its first transmitted Sync message (SyncSN not 0) then this corresponds to a previous synchronization process. For this reason the router takes a snapshot with the corresponding sequence number (MySnapshotSN). The router discovers the BootTime of the neighbor through the received Sync message. The router declares itself as the Master of the synchronization process, placing its neighbor in SLAVE state and transmits a new Sync message to it, with information regarding all known sequence numbers (BootTime of both routers and MySnapshotSN), SyncSN set to 0 and Master flag set.

When a newer Sync message is transmitted both the NeighborLivenessTimer and the SyncRetransmissionTimer are set, in order to detect failures during the synchronization and retransmit a Sync message in case no proper response is obtained. When the neighbor is placed in SYNCED state, the SyncRetransmissionTimer is canceled and the NeighborLivenessTimer must be set according to the Hello HoldTime transmitted in the Sync message or a default value is set;.

3) When the SyncRetransmissionTimer expires, it means that no proper response was obtained in due time. For this reason, the last transmitted Sync message is retransmitted and this timer is reset.

4) When the NLT expires, it means that the neighbor is considered to have failed. This can happen in an ongoing synchronization process or after this process has finished correctly. In both cases, the neighbor is placed in UNKNOWN state and all information about it is removed.

5) If an Hello message is received from a neighbor that is in SYNCED state, the NLT is reset, according to the value present in the received message. Reception of Hello messages during a synchronization do not cause a reset of this timer, because progress in the synchronization process is what defines the neighbor to be alive.

All trees received during the current synchronization process must be stored. All stored state is only valid after the neighbor router finishes successfully this synchronization process (neighbor transitions to the SYNCED state).

6 Neighbor state definition regarding (S,G) Tree

For a given neighbor, a router must monitor two types of information for each (S,G) tree:

- Upstream information - Whether the neighbor declares itself to be Upstream regarding the (S,G) tree. Two states are possible: UPSTREAM or NOT UPSTREAM;
- Interest information - Whether the neighbor is interested or not in receiving data packets regarding the (S,G) tree. Two states are possible: INTERESTED or NOT INTERESTED.

For a neighbor that declares itself as UPSTREAM, it is also required to store its RPC to source S, in order to elect the router responsible for forwarding multicast data packets in a given link. Also, in the case of a neighbor declaring itself as UPSTREAM, this also means that the neighbor is NOT INTERESTED.

Absence of upstream information, regarding a neighbor, must be considered as NOT UPSTREAM. Absence of interest information, regarding a neighbor, can be manually configured to be considered as INTERESTED or NOT INTERESTED, in order to allow or prevent the initial flooding behavior.

The maintenance of a neighbor state is controlled by the reception of control messages such as IamUpstream, IamNoLongerUpstream, Interest, NoInterest and Sync. The first four messages include a SN in order to filter messages that may arrive out of order and also to not reinterpret messages that have already been processed. So when one of these control messages is received, it is required to verify if it is considered to be "fresh". The Sync messages will exchange initial state between two routers, and state regarding a router must only be considered as valid when this synchronization successfully finishes. When a synchronization finishes successfully, we have the guarantee that the information exchanged in Sync messages is correct.

In event 1 of Table 2, if a router receives a "fresh" IamUpstream message from a given neighbor N, this means that N considers itself to be Upstream. The router reacts to this message by changing the state of N, in order to consider it as being UPSTREAM. The RPC included in this message must also be stored in order to determine the router responsible for forwarding data packets on the link. Since N is considered to be UPSTREAM, this implicitly means that it is not interested in receiving data packets, being this information also stored.

In event 2 of Table 2, if a router receives a "fresh" IamNoLongerUpstream message from a given neighbor N, this means that N no longer considers itself to be Upstream. The router reacts to this message by setting N as NOT UPSTREAM regarding this tree. Regarding the interest, the router can set the interest of N as the default/omission interest. The neighbor would then transmit an Interest or NoInterest if the router that received the IamNoLongerUpstream is the one responsible for forwarding multicast traffic to the link, which would cause a correct set of its interest state.

In event 3 of Table 2, if a router receives a "fresh" Interest message from a given neighbor N, this means that N is NOT UPSTREAM and is INTERESTED in receiving data packets.

In event 4 of Table 2, if a router receives a "fresh" NoInterest message from a given neighbor N, this means that N is NOT UPSTREAM and is NOT INTERESTED in receiving data packets.

	Event	Action
1	Receive IamUpstream(S,G,RPC) with SN and BT from neighbor N in interface I and shouldProcess(S,G,N,SN,BT)	Set N as UPSTREAM regarding (S,G) with RPC Set N as NOT INTERESTED regarding (S,G)
2	Receive IamNoLongerUpstream(S,G) with SN and BT from neighbor N in interface I and shouldProcess(S,G,N,SN,BT)	Set N as NOT UPSTREAM regarding (S,G)
3	Receive Interest(S,G) with SN and BT from neighbor N in interface I and shouldProcess(S,G,N,SN,BT)	Set N as NOT UPSTREAM regarding (S,G) Set N as INTERESTED regarding (S,G)
4	Receive NoInterest(S,G) with SN and BT from neighbor N in interface I and shouldProcess(S,G,N,SN,BT)	Set N as NOT UPSTREAM regarding (S,G) Set N as NOT INTERESTED regarding (S,G)
5	Receive Sync message from neighbor N that included information about (S,G), with the corresponding RPC, from an ongoing synchronization process and no fresher state is stored	Set N as UPSTREAM regarding (S,G) with RPC Set N as NOT INTERESTED regarding (S,G)
6	Neighbor N fails or N is resynchronizing	Remove previously stored upstream and interest information about N

Table 2: Maintenance of neighbor state regarding a given (S,G) tree.

Event 5 of Table 2 represents the reception of Sync messages from a neighbor that it is in an ongoing synchronization process with the own router. If the synchronization has already finished, following Sync messages referring the same synchronization process (same BootTimes and SnapshotSNs) must be ignored (in terms of storage of state). If the synchronization has not finished yet, and the Sync message is from an ongoing synchronization that is making progress, then tree information can only be stored if fresher state was not stored in the meantime (this can happen if control messages transmitted during the synchronization have already been processed regarding the neighbor router). In order to verify if more recent information regarding a given (S,G) tree, included in a Sync message, has already been stored, the router compares the SnapshotSN of the ongoing synchronization process with the SN of the last received non-Sync message regarding (S,G). If the router already stores a SN greater than the neighbor's SnapshotSN, the router does not do anything (because it already stores fresher information), otherwise if there is no SN stored about (S,G) then N is considered as UPSTREAM (because the Sync message is transmitting fresher information).

In event 6 of Table 2, if a router fails or if it triggers a new synchronization process, all stored upstream and interest state about it is removed. In case of a failure, the neighbor by no longer participating in the multicast routing process, all its state can be safely removed. In case of a new synchronization with a known neighbor, all stored state, before the new synchronization starts can be safely removed, since fresher state will be exchanged in new Sync messages.

The shouldProcess referred in Table 2 corresponds to a function used to determine whether a message must be processed. The following pseudocode shows which tests are used to determine if a received message is considered "fresh ". If shouldProcess returns True, the message must be processed and the state of the neighbor must be set, otherwise the message must be ignored and the corresponding neighbor's state not changed.

```
function shouldProcess(S,G,N,SN,BT):                                    1
If ''Neighbor N is UNKNOWN or N.CurrentSyncSN==0'':                      2
    Return False                                                         3
Else If SN<=N.NeighborSnapshotSN or BT!=N.BootTime or                    4
        SN<=N.CheckpointSN:                                              5
    Return False                                                         6
Else If SN==N.LastSN(S,G):                                               7
    Send ACK(S,G,SN) to N                                                8
    Return False                                                         9
Else If SN > N.LastSN(S,G):                                              10
    N.LastSN(S,G)=SN                                                     11
```

```
    Send ACK(S,G,SN) to N                                    12
    Return True                                              13
Else:                                                        14
    Return False                                             15
```

Listing 2: Pseudocode used to determine "fresh" messages.

Regarding Listing 2, the first test to determine if a message should be processed (line 2) is if the source of the transmitted message is already considered to be a neighbor. If the neighbor is in UNKNOWN state, the message must be ignored, and possibly trigger a synchronization with it. If the router is in an ongoing synchronization process with it (neighbor in MASTER or SLAVE state), then the message must not be processed if the SyncSN of the current synchronization is 0. In case the synchronization is currently with a SyncSN equal to 0, then there is no guarantee that the exchanged BootTimes and SnapshotSNs are correct.

If the neighbor is in SYNCED state or if it is in an ongoing synchronization process with the CurrentSyncSN greater than 0, then there is the guarantee that the exchanged BootTimes and SnapshotSNs were reliably exchanged. In that case, a message must not be processed if the NeighborSnapshotSN or CheckpointSN are higher than the SN received in this message or if the BootTime is different from the one obtained during the synchronization process (lines 4-5). If the NeighborSnapshotSN is greater than the SN, this means that the synchronization process already exchanged fresher state, compared to the state included in this message. If the CheckpointSN is greater than the received SN, then the received message corresponds to a retransmission which has already been acknowledged by the own router. If the BootTime is different from the one obtained during the synchronization process, this means that the message was transmitted before or after a reboot. In both cases the message must not be processed, but in case the received BootTime is greater than the one stored, a new synchronization with this neighbor should be triggered, since the stored state regarding the router may be inconsistent.

If the previous two test conditions were not true for the received message, then it may correspond to a simple retransmission of already stored state or may include fresher state that was not stored yet. If the last SN received from this neighbor, regarding the tree that this message is referring to, is equal to the message's SN (line 7), this represents a retransmission of a message that has already been processed. It may happen that the retransmission is occurring due to the loss of the acknowledgment, so in this case a ACK must be retransmitted and the content of the message should not be processed.

If the previous three test conditions were not true for the received message, then it may correspond to fresher state that was not stored yet. The message must only be processed if the received SN is greater than the last received SN regarding the same tree (line 10). In this case an ACK must be transmitted to the neighbor router, the SN must be stored as the most recent SN received regarding this tree and the message processed.

In any other case (line 14) the packet must be ignored, since it does not have fresher state compared to the one already stored by the router.

Required calculations based on stored state.

Based on the stored state regarding each neighbor, for all trees, an interface must determine which neighbor is considered to be UPSTREAM that offers the lowest RPC. To this neighbor we will call it BestUpstreamNeighbor. This can change whenever an interface receives any type of control messages, new neighbors appear on the network and known neighbors fail.

In order to facilitate the explanation of the following subsections, we define two sets of BestUpstreamNeighbors, for a given (S,G) tree:

- BURI - The BestUpstreamNeighbor of the (S,G) tree and its corresponding RPC, that is connected to the root interface, i.e. neighbor connected to the root interface that is considered to be UPSTREAM and that offers the lowest RPC, regarding the (S,G) tree;
- LBUNI - List of BestUpstreamNeighbors connected to non-root interfaces and their corresponding RPC, regarding the (S,G) tree. This list includes all non-root interfaces UPSTREAM neighbors, that offer the lowest RPC on their corresponding interface.

As we can see, the BURI and LBUNI are defined depending on the root and non-root interfaces. This means that changes of interface roles or the addition/removal of interfaces can change the BURI and LBUNI structures.

Also, changes of upstream state in neighbors can modify these structures, like the reception of any control message, the failure of neighbors that were considered to be UPSTREAM and also the discovery of new neighbors that are considered to be UPSTREAM.

If the BestUpstreamNeighbor of any interface suffers an RPC change this can modify the corresponding BURI/LBUNI structures.

In case there is no reachability regarding the source of a tree, due to the absence of entries in the unicast routing table regarding the subnet of the source, this means that there is no root interface. For this reason, all interfaces are considered to be non-root. This implies that the BURI must be empty.

Please note that if the BURI or LBUNI is empty or not set, this is because the router does not connect to any UPSTREAM neighbor in its root or non-root interfaces, respectively.

The BURI and LBUNI structures are very important because they define which directly connected neighbors offer the lowest RPC to the source of multicast traffic. This allow us to determine if the tree is built correctly and also if an interface is responsible for forwarding multicast traffic.

7 (S,G) Tree definition

A router to maintain state regarding a given tree, it must be logically connected to the source of multicast traffic. The originator will maintain state regarding a tree as long as the source keeps transmitting multicast data packets. A non-originator router will maintain state of a tree as long as there are UPSTREAM neighbors connected to it.

Since the state of a tree depends whether the source is or is not directly connected to a router, we will distinguish the state machines of originator and non-originator routers.

7.1 (S,G) Tree definition state machine of Originator router

This state machine controls the state of a given (S,G) tree, in which the source is directly connected to the router.

The state of a tree for originator routers depends on the reception of data packets from the directly connected source and also on the existence of UPSTREAM neighbors connected to non-root interfaces.

Trees of originator routers require a timer, called Source Active Timer, used to control the state of a tree. When a data packet is received from a directly connected source, the timer is set and the tree will keep in ACTIVE state as long as the timer does not expire. When this timer expires, the tree is no longer considered to be ACTIVE, due to the absence of data packets from the source.

Table 3 shows how originator routers determine the state of a tree.

		ACTIVE	**UNSURE**	**INACTIVE**
1	Receive (S,G) data packets in root interface	Set SAT	→ ACTIVE Set SAT	→ ACTIVE Set SAT
2	SAT expires and LBUNI is non-empty	→ UNSURE	NA	NA
3	SAT expires and LBUNI is empty	→ INACTIVE	NA	NA
4	Received IamUpstream regarding (S,G) tree or new neighbor finishes synchronization and is considered as Upstream regarding (S,G) causing LBUNI to get non-empty			→ UNSURE
5	Neighbor fails or new synchronization with known neighbor is triggered or received IamNoLongerUpstream(S,G) causing LBUNI to get empty		→ INACTIVE	

Table 3: Maintenance of (S,G) tree state of Originator router.

In event 1 of Table 3, the reception of data packets from the root interface, i.e. interface that directly connects to the source, means that the tree is in ACTIVE state. This causes the SAT to be set. The tree will keep in ACTIVE state while the SAT does not expire.

In event 2 and 3 of Table 3, the expiration of the SAT means that the tree is no longer ACTIVE due to the absence of data packets from the source. If the router connects to UPSTREAM neighbors via

one of its non-root interfaces (event 2), this means that the tree transitions to UNSURE state, otherwise (event 3) the tree transitions to INACTIVE state. If the router transitions to UNSURE state, this means that there are other routers that can connect to the source, due to the existence of redundant paths.

In event 4 of Table 3, if the tree was in INACTIVE state, this means that the router did not consider the source to be "active" and it was not connected to any UPSTREAM neighbor. If the tree was in INACTIVE state and the router receives an IamUpstream message from a non-root interface, causing the router to be connected to at least one neighbor UPSTREAM in one of its non-root interfaces, this causes the tree to transition to UNSURE state. The same thing can occur when a synchronization finishes successfully with a neighbor router that is connected to a non-root interface, and that neighbor is considered to be UPSTREAM regarding (S,G).

In event 5 of Table 3, if the tree was in UNSURE state, this means that the router did not consider the source to be "active" and it was connected to at least one UPSTREAM neighbor in one of its non-root interfaces. If due to a failure of a neighbor, removal of an interface or reception of a control message that causes this router to no longer be connected to UPSTREAM neighbors in any of its non-root interfaces, this causes the router to transition to INACTIVE state. If a new synchronization is triggered with a known neighbor, all its state is ignored during this process, i.e. the router is not considered as UPSTREAM during a synchronization.

The definition of the tree state is important to determine which type of control messages an interface must transmit (explained in the next sections).

7.2 (S,G) Tree definition state machine of Non-Originator router

This state machine controls the state of a given (S,G) tree, in which the source is not directly connected to the router.

The state of a tree for a non-originator router depends solely on the existence of UPSTREAM neighbors connected to the router's interfaces and also on the RPC of the own router and of its UPSTREAM neighbors.

The (S,G) tree will be in:

- ACTIVE state - if BURI is non-empty and it offers a lower RPC compared to the own router's RPC to source S;
- UNSURE state - if BURI is empty and LBUNI is non-empty or BURI is non-empty and it offers an RPC that is greater or equal to the own router's RPC to source S;
- INACTIVE state - if BURI is empty and LBUNI is empty.

If the router is connected to an UPSTREAM neighbor via its root interface and a loop is impossible to exist (RPC offered by the BestUpstreamNeighbor connected to the root interface is lower than the own router' RPC), this means that the tree is in ACTIVE state.

If the router is not connected to UPSTREAM neighbors in its root interface (BURI is empty) but connects to at least one UPSTREAM neighbor in one of its non-root interfaces (LBUNI non-empty), this means that the tree is in UNSURE state. The same UNSURE state applies when there are UPSTREAM neighbors connected to the router's root interface (BURI non-empty) but there is the possibility of a loop (RPC of the BestUpstreamNeighbor not lower than the own router's RPC).

If the router does not connect to UPSTREAM neighbors in any of its interfaces (BURI and LBUNI empty), the tree is in INACTIVE state.

The maintenance of UPSTREAM neighbors is very important and the tree's state should be rechecked whenever the BURI and/or LBUNI structures are manipulated. This manipulation may be due to the addition or removal of UPSTREAM neighbors, that can change the state of a tree. Changes regarding the root interface can also change the state of a tree, due to the existence/absence of UPSTREAM neighbors connected to the new root interface.

The RPC of the own router and of the BestUpstreamNeighbor connected to the root interface (BURI) is also important in the determination of a tree state. For this reason, any change to the RPC of the BestUpstreamNeighbor or of the own router, must trigger the verification of the new tree state.

When the interface's roles change, i.e. the root interface changes, the tree state must also be rechecked, since the new root interface may or may not be connected to UPSTREAM neighbors (changes to BURI and LBUNI).

The definition of the tree state is important to determine which type of control messages an interface must transmit (explained in the next sections).

7.3 (S,G) Non-Originator becomes Originator router

If a non-originator router has a new interface that directly connects to the source of a tree that was already being maintained, the router becomes originator.

If the router was non-originator and considered the tree as being ACTIVE, when it becomes originator, the tree should remain ACTIVE even if no data packet was received by the new root interface. This is to not trigger the removal of the tree. In this case, the router sets the Source Active Timer, in order to control when it should transition from ACTIVE to UNSURE or INACTIVE states. This is because if the router was previously ACTIVE then there was already a redundant path that connected this router to the source of multicast traffic, through another originator router, and if it was ACTIVE then the source was transmitting data packets.

If the router was UNSURE (BURI empty and LBUNI non-empty), it should remain UNSURE until data packets are received through the new root interface.

From this point on, the calculation of the tree state will be performed using the "Originator tree state definitions" (in section 7.1).

7.4 (S,G) Originator becomes Non-Originator router

If a originator router disables its root interface, that directly connected to the source of multicast traffic, it becomes a non-originator router if it is no longer directly connected to the source.

This means that the tree must now be recalculated not considering the reception of data packets, through the new root interface. This calculation must be performed only by verifying the existence of UPSTREAM neighbors connected to the new root and non-root interfaces (BURI and LBUNI), like referred in section 7.2.

8 Assert state of (S,G) regarding Non-Root interfaces

When two or more non-root interfaces connect to the same link, only one of them requires to transmit data packets in case there are routers interested in receiving those data packets, otherwise there will be duplication of data packets.

The Assert state machine has two states:

- **ASSERT WINNER (AW)** - interface is responsible for forwarding data packets to a given link;
- **ASSERT LOSER (AL)** - interface is not responsible for forwarding data packets to a given link.

The interface will be in an AW state, when:

- Tree is in ACTIVE state, and
- The interface offers the best RPC to source S, compared to all UPSTREAM neighbors connected to the interface. In case of a tie, the IP will be used to break the tie, being the greatest IP the winner

OR

- Tree is in UNSURE state, and
- No UPSTREAM neighbor is connected to the interface

OR

- Tree is in INACTIVE state

The first condition means that the tree is ACTIVE and the interface by offering the best reachability to source S must be responsible for forwarding data packets to the link.

In the second and third conditions, the tree is not ACTIVE, meaning that there is no logical connection between the root interface of a router with the source of multicast traffic. If the non-root interface does not connect to UPSTREAM neighbors (no BestUpstreamNeighbor connected to this non-root interface), there are no routers connected to the same link that logically connect to the source. In order to be able to forward the first multicast data packets that might arrive at the root interface before information

regarding UPSTREAM neighbors is known, the non-root interface should be in AW state. If the flood of initial data packets is not intended, the default interest information will be used to control this behavior.

Interfaces in AL state will not forward any multicast data packets, regardless of their neighbors' interests.

9 Downstream Interest in (S,G) at Non-Root interfaces

A non-root interface must determine if there is downstream interest of neighbors or directly connected hosts.

The detection of interest from neighbors is through their interest state (INTERESTED or NOT INTERESTED).

The detection of interest from hosts is through an independent protocol used for this matter. IGMP is used in IPv4 networks, while MLD is used for IPv6 networks.

A non-root interface can be in one of two states regarding downstream interest:

- **DOWNSTREAM INTERESTED** - there is interest through this non-root interface;
- **NOT DOWNSTREAM INTERESTED** - there is no interest through this the non-root interface.

A non-root interface is DOWNSTREAM INTERESTED when:

- At least one neighbor has its interest state set to INTERESTED, OR
- IGMP/MLD define this tree in a state where there is interest from directly connected hosts.

Or is NOT DOWNSTREAM INTERESTED otherwise.

This state machine must be reverified in reaction to changes at IGMP/MLD, due to the join/leave of hosts, and also in reaction to changes of interest of neighbors, their removal or their addition.

Initially an interface may not know the interest state of all its neighbors. All neighbors that are NOT UPSTREAM and that have not set their interest state, are initially considered as INTERESTED or NOT INTERESTED (configurable), until the router receives their corresponding interest.

This is used to set the initial state of the Downstream Interest state machine. This will be used to control the initial flooding behavior of the protocol.

By considering that all neighbors are initially in INTERESTED state (of the ones that have not transmitted yet their interest message), this can be used to flood initial data packets, in order for these to not be lost.

By considering that all neighbors are initially NOT INTERESTED (of the ones that have not transmitted yet their interest message), this can be used to prevent the flooding behavior of the multicast routing protocol. This will cause the first data packets to be lost, since all neighbors are initially not interested in receiving data packets.

10 Forwarding of (S,G) data packets

A non-root interface is responsible for forwarding multicast data packets that arrive at the root interface. Depending on the Assert state and on the Downstream Interest state, a non-root interface may or may not forward those data packets.

A non-root interface can be in one of two states, regarding the forwarding of data packets:

- **FORWARDING** - the non-root interface must forward data packets;
- **PRUNED** - the non-root interface must not forward data packets.

A non-root interface will be in a FORWARDING state when:

- The interface is in AW state, and
- The interface is in DOWNSTREAM INTERESTED state, and
- The interface is not directly connected to the source.

Or in PRUNED state otherwise.

As we can see, the Forwarding state of a non-root interface depends on the Assert and Downstream Interest state machines and whether the interface is connected directly to the source. For this reason, whenever there is a change in any of these conditions, the forwarding decision must be rechecked.

The condition of being directly connected to the source is required in order to prevent loops in the link that is connected to the source. In case a router connects directly to the source link by two interfaces, one would be root and the other would be non-root. The interface that is non-root must not forward packets regardless of its assert and downstream interest state.

As we can see, by controlling the interest state of neighbors that have not transmitted yet their interest message, the router determines its initial Downstream Interest state. Interfaces not directly connected to the source by being in AW and DOWNSTREAM INTERESTED states, forward data packets. By configuring the initial Downstream Interest, this allows or prevents the flooding behavior.

11 Router/Root interface interest in (S,G) data packets

Depending on the forwarding state of non-root interfaces, a router may be interested in receiving data packets from its root interface.

There are two states in which a router/root interface can be at:

- **INTERESTED** - router is interested in receiving data packets because it must forward them to at least one non-root interface;
- **NOT INTERESTED** - router is not interested in receiving data packets because no non-root interface will forward them.

A router will be in INTERESTED state when:

- At least one non-root interface is in a FORWARDING state.

And NOT INTERESTED, otherwise (all non-root interfaces are in a PRUNED state).

This state will control the type of interest message that a router will transmit through its root interface, in order to notify the router responsible for forwarding multicast traffic about its interest.

12 Control message transmission regarding (S,G)

A router is responsible for notifying its neighbors about its state:

- Whether it is UPSTREAM or NOT UPSTREAM regarding a given tree;
- If NOT UPSTREAM, if it is INTERESTED or NOT INTERESTED in receiving data packets regarding a given tree.

In this subsection, we define when each type of control messages must be transmitted by a given interface type.

12.1 Control message transmission regarding (S,G) from Root interfaces of Originator routers

Root interfaces of originator routers do not transmit any type of control message, since they are directly connected to the source of multicast data packets.

12.2 Control message transmission regarding (S,G) from Root interfaces of Non-Originator routers

Root interfaces of non-originator routers can transmit three types of control messages, which are the IamNoLongerUptream, Interest and NoInterest. These control messages can notify respectively, that a router is no longer considered to be UPSTREAM and if it is INTERESTED or NOT INTERESTED in receiving data packets.

	Event	Action
1	Interface role changes (Non-Root→Root) and tree remains in ACTIVE state	Send IamNoLongerUpstream(S,G) Send Interest (if INTERESTED) or NoInterest (if NOT INTERESTED) to BestUpstreamNeighbor
2	Interface role changes (Non-Root→Root) and tree was UNSURE and transitions to ACTIVE state	Send Interest (if INTERESTED) or NoInterest (if NOT INTERESTED) to BestUpstreamNeighbor
3	Interface role changes (Non-Root→Root) and tree was ACTIVE and transitions to non ACTIVE state (UNSURE or INACTIVE) and BestUpstreamNeighbor is null	Send IamNoLongerUpstream(S,G)
4	Interface role changes (Non-Root→Root) and Tree was ACTIVE and transitions to UNSURE and BestUpstreamNeighbor≠null (possible loop detected)	Send IamNoLongerUpstream(S,G) Send Interest (if INTERESTED) or NoInterest (if NOT INTERESTED) to BestUpstreamNeighbor
5	Interface role changes (Non-Root→Root) and tree remains UNSURE and BestUpstreamNeighbor≠null	Send Interest (if INTERESTED) or NoInterest (if NOT INTERESTED) to BestUpstreamNeighbor
6	Interface role does not change and BestUpstreamNeighbor changes and is not null	Send Interest (if INTERESTED) or NoInterest (if NOT INTERESTED) to BestUpstreamNeighbor
7	Interface role does not change and BestUpstreamNeighbor does not change and router's interest changes (NOT INTERESTED→INTERESTED or INTERESTED→NOT INTERESTED)	Send Interest (if INTERESTED) or NoInterest (if NOT INTERESTED) to BestUpstreamNeighbor
8	Interface role does not change and receive IamUpstream from current BestUpstreamNeighbor and BestUpstreamNeighbor does not change	Send Interest (if INTERESTED) or NoInterest (if NOT INTERESTED) to BestUpstreamNeighbor

Table 4: Message transmission through Root interface of non-originator routers.

The transmission of IamNoLongerUpstream can happen when interfaces change roles, being the new root interface an interface that has previously notified its neighbors that it was considered to be UPSTREAM. Since the interface is now root type, it is required to inform its neighbors that this interface is no longer considered to be UPSTREAM, which is accomplished by transmission of this type of message.

The transmission of Interest or NoInterest messages are destined to the BestUpstreamNeighbor, which is the UPSTREAM neighbor responsible for forwarding data packets to the link that connects the root interface. The transmission of these types of messages can happen when the router changes interest or when the BestUpstreamNeighbor changes. The type of message that is transmitted depends whether the own router is INTERESTED or NOT INTERESTED.

Table 4 summarizes all events and corresponding actions, regarding the transmission of control messages through the root interface of a non-originator router.

In event 1 of Table 4, when interfaces change roles and the tree remains in an ACTIVE state, due to the existence of an UPSTREAM neighbor connected to the new root interface with a lower RPC than the own router's RPC, this interface must inform all its neighbors that it is no longer considered to be UPSTREAM. This is accomplished by the transmission of an IamNoLongerUpstream message. Also, the new root interface, by being an interface that receives data packets and forwards to non-root interfaces, requires to inform the BestUpstreamNeighbor about its interest regarding (S,G) data packets via an Interest or NoInterest message.

In event 2 of Table 4, when interfaces change roles and the tree transitions from UNSURE to ACTIVE state, the new root interface requires to inform the BestUpstreamNeighbor about its interest regarding (S,G) data packets via an Interest or NoInterest message. This event is caused by both changes in the unicast routing table and also due to the presence of UPSTREAM neighbors connected to the new root interface which offer a lower RPC compared to the own router's RPC. In case of this event, it is not required to transmit an IamNoLongerUpstream, since all neighbors already know that this router is NOT

UPSTREAM, because the tree was previously in UNSURE state.

In event 3 and 4 of Table 4, when interfaces change roles that cause the tree to transition from an ACTIVE state to UNSURE state, the new root interface requires to first inform all neighbors that it is no longer considered to be UPSTREAM. The same is required when the router transitions from ACTIVE to INACTIVE state. This is accomplished by the transmission of an IamNoLongerUpstream message. This transmission is required since the tree was previously in an ACTIVE state and the new root interface, which was an old non-root interface, informed its neighbors that it was considered to be UPSTREAM. If the new root interface does not connect to UPSTREAM neighbors (event 3), then no additional action is required. If the new root interface connects to UPSTREAM neighbors, but none of them offer a RPC lower than the own router's RPC, then it is possible that a loop exists in the network and for this reason the tree is considered to be in UNSURE state (event 4). In this last case, since there are UPSTREAM neighbors connected to the root interface, the router still requires to inform the UPSTREAM neighbor that offers the lowest RPC, about its interest regarding the reception of multicast data packets, via an Interest or NoInterest message.

In event 5 of Table 4, when interfaces change roles, the tree remains in UNSURE state and the new root interface connects to UPSTREAM neighbors, an interest message must be transmitted to the router that offers the lowest RPC. That router, the BestUpstreamNeighbor, will make a forwarding decision based on the interest of all neighbors, thus requiring the new root interface to transmit to it its intent.

Event 6 of Table 4, describes what a root interface must do when the router responsible for forwarding multicast data, the BestUpstreamNeighbor, changes. This can happen due to a failure of the previous BestUpstreamNeighbor or reception of IamUpstream with a lower RPC from the new BestUpstream-Neighbor or reception of IamUpstream with a greater RPC from the previous BestUpstreamNeighbor or reception of IamNoLongerUpstream from the previous BestUpstreamNeighbor or there was no previous BestUpstreamNeighbor. When this happens, the new BestUpstreamNeighbor must be informed about the interest of this router, via the transmission of an Interest or NoInterest message.

In Table 4, when there is no change of the BestUpstreamNeighbor but the router changes its interest regarding the reception of multicast data packets (event 7), the root interface must inform about its new interest to the BestUpstreamNeighbor, via an Interest or NoInterest message, if the router transitions respectively to INTERESTED or NOT INTERESTED state. This can happen due to a change of forwarding state in all or some of the router's non-root interfaces.

Event 8 of Table 4, describes what a root interface must do when the current BestUpstreamNeighbor transmits an IamUpstream message, not causing a change of the BestUpstreamNeighbor. Since there is the possibility that the BestUpstreamNeighbor lost the interest state of this interface, it is required to retransmit an interest message to it.

In the case of the actions that require to transmit two messages, the order of these transmissions must be respected in order to have an interest message with a greater sequence number.

12.3 Control message transmission regarding (S,G) from Non-Root interfaces

Non-root interfaces of any router, originator or non-originator, can transmit three types of control messages, which are the IamUpstream, IamNoLongerUpstream and NoInterest. These control messages can notify respectively, that a router considers itself to be UPSTREAM, NOT UPSTREAM or NOT UPSTREAM and NOT INTERESTED in receiving data packets regarding a given tree.

The transmission of IamUpstream can happen when the tree transitions to an ACTIVE state or when the tree was already in an ACTIVE state but the router suffers an RPC change. This message will notify all neighbors connected to the interface, that this router considers itself to be UPSTREAM, with a given RPC.

The transmission of IamNoLongerUpstream can happen when the tree was ACTIVE and transitions to an UNSURE or INACTIVE state. This requires to notify all neighbors that this router no longer considers itself to be UPSTREAM regarding a given tree.

The transmission of a NoInterest message is destined to the BestUpstreamNeighbor and can happen when the tree is in UNSURE state and there are UPSTREAM neighbors connected to the non-root interface. Since a non-root interface is just responsible for forwarding data packets, not receiving them, and by not being logically connected to the source via the root interface, this router requires to inform the BestUpstreamNeighbor connected to it about its disinterest.

Table 5 summarizes all events and corresponding actions, regarding the transmission of control messages through a non-root interface.

	Event	Action
1	Interface role does not change and interface is not directly connected to the source and tree transitions to ACTIVE state (INACTIVE→ACTIVE or UNSURE→ACTIVE)	Send IamUpstream(S,G,MyRPC)
2	Interface role changes (Root→Non-Root) and interface is not directly connected to the source and tree remains or transitions to ACTIVE state	Send IamUpstream(S,G,MyRPC)
3	Interface role does not change and tree state ACTIVE→UNSURE and BestUpstreamNeighbor==null	Send IamNoLongerUpstream(S,G)
4	Tree state ACTIVE→INACTIVE	Send IamNoLongerUpstream(S,G)
5	Interface role does not change and tree state ACTIVE→UNSURE and BestUpstreamNeighbor≠null	Send IamNoLongerUpstream(S,G) Send NoInterest(S,G) to BestUpstreamNeighbor
6	Interface role changes (Root→Non-Root) and tree state ACTIVE→UNSURE and BestUpstreamNeighbor≠null	Send NoInterest(S,G) to BestUpstreamNeighbor
7	Tree remains in UNSURE state and receive IamUpstream from BestUpstreamNeighbor and BestUpstreamNeighbor does not change	Send NoInterest(S,G) to BestUpstreamNeighbor
8	Tree remains in UNSURE state and receive IamUpstream or IamNoLongerUpstream or neighbor fails and BestUpstreamNeighbor changes and BestUpstreamNeighbor≠null	Send NoInterest(S,G) to BestUpstreamNeighbor
9	Tree state INACTIVE→UNSURE due to reception of IamUpstream and BestUpstreamNeighbor≠null	Send NoInterest(S,G) to BestUpstreamNeighbor
10	Interface is not directly connected to the source and Tree remains in ACTIVE state and MyRPC changes	Send IamUpstream(S,G,MyRPC)

Table 5: Message transmission through Non-Root interface.

In event 1 of Table 5, when the state of a tree transitions to ACTIVE, not due to an interface role change, the router must inform its neighbors that it is UPSTREAM regarding that tree. This is accomplished by the transmission of an IamUpstream message through non-root interfaces. The transition to ACTIVE state is due to the existence of UPSTREAM neighbors connected to the root interface of a router, which happens when the router hears for the first time an IamUpstream message through the root interface (with the neighbor RPC lower than the own router's RPC).

In event 2 of Table 5, when the router suffers an interface role change causing a transition to an ACTIVE state, the router must inform its neighbors that it is UPSTREAM regarding this tree. This is accomplished by the transmission of an IamUpstream message through all non-root interfaces. If the router remains in ACTIVE state after the interface role changes, the new non-root interface must inform its neighbors that it is UPSTREAM, via the transmission of an IamUpstream message.

In event 3 of Table 5, when the state of a tree transitions from ACTIVE to UNSURE and there are no UPSTREAM neighbors connected to the non-root interface, the interface must inform its neighbors that it is no longer considered as UPSTREAM regarding that tree. This is accomplished by the transmission of an IamNoLongerUpstream message. This transition happens due to the absence of UPSTREAM neighbors or data packets on the root interface, depending whether the router is originator or non-originator, but the router is still connected to UPSTREAM neighbors via one of its non-root interfaces.

In event 4 of Table 5, when the state of a tree transitions from ACTIVE to INACTIVE, the interface must inform its neighbors that it is no longer considered as UPSTREAM regarding that tree. This is accomplished by the transmission of an IamNoLongerUpstream message. This transition happens when the router no longer connects to UPSTREAM neighbors and/or no longer receives data packets from the source, depending on whether the router is originator or non-originator.

In event 5 and 6 of Table 5, when the state of a tree transitions from ACTIVE to UNSURE, all its neighbors must not consider the router as UPSTREAM. If the interface role did not change, i.e. the interface is still non-root type, the router must first transmit an IamNoLongerUpstream message. If the non-root interface connects to UPSTREAM neighbors, the interface must inform the BestUpstreamNeighbor that it is not interested in receiving data packets, because it is a non-root interface. This is accomplished by the transmission of a NoInterest message destined to the BestUpstreamNeighbor,

regardless of whether this interface changed or not its role.

In event 7 of Table 5, when the state of a tree remains in UNSURE state and the BestUpstreamNeighbor transmits an IamUpstream message that does not cause a change of the BestUpstreamNeighbor, the interface must retransmit a NoInterest message to it. This is performed because there is no guarantee that the BestUpstreamNeighbor is still maintaining interest state of this non-root interface, thus a NoInterest message is transmitted to it.

In event 8 of Table 5, when the state of a tree remains in UNSURE state and there is still an UPSTREAM neighbor connected to the interface and the BestUpstreamNeighbor changes, due to a failure or reception of IamUpstream or IamNoLongerUpstream messages, the router requires to inform the BestUpstreamNeighbor that it is not interested in receiving data packets, via a NoInterest message.

In event 9 of Table 5, when the state of a tree transitions from an INACTIVE to UNSURE state and there is an UPSTREAM neighbor connected to the non-root interface, which will be the BestUpstreamNeighbor, the router must inform it that it is not interested in receiving data packets, via the transmission of a NoInterest message. This can happen when there was no UPSTREAM neighbors connected to any interface and a non-root interface receives an IamUpstream message.

In event 10 of Table 5, when the state of a tree remains in ACTIVE state and the router suffers an RPC change, the non-root interface must notify all neighbors about this change. This is accomplished by the transmission of an IamUpstream message with the new RPC. This is required in order to reelect the router responsible for forwarding data packets.

In the case of the actions that require to transmit two messages, the order of these transmissions must be respected in order to have an interest message with a greater sequence number.

13 Control message reliability

Regarding Sync messages, these are reliably protected by a Stop-and-Wait mechanism, which has already been addressed in section 5.

Control messages such as IamUpstream, IamNoLongerUpstream, Interest and NoInterest are reliably protected using an ACK-protection mechanism, meaning that those messages must be acknowledged by the corresponding receiver(s). The transmission of ACK messages, in reaction to the reception of control messages has already been addressed in Appendix 6.

The reliability of transmitted control messages will be addressed in this section.

First of all, IamUpstream and IamNoLongerUpstream messages are destined to all neighbors attached to a link (destination IP address is multicast). This means that in order to consider that these messages have been reliably transmitted, an interface must receive the ACK from all known neighbors attached to that link. Regarding Interest and NoInterest messages, by being destined to a single router (destination IP address is unicast), the interface that transmitted those messages requires to wait for the reception of the ACK from the corresponding destination.

An interface must wait a given amount of time to allow its neighbor(s) to acknowledge the message. If after a given amount of time, the interface did not get the ACK from at least one neighbor that was supposed to acknowledge it, the message must be retransmitted. This retransmission will occur until all supposed neighbors ACK the message or the ones that have not acknowledged yet are declared to have failed. For this reason, a timer called RetransmissionTimer is used, that is reset whenever a newer message or retransmission occurs, for messages regarding a given interface and a given tree.

When a new multicast message is transmitted (IamUpstream or IamNoLongerUpstream), regarding a given (S,G) tree, all messages that refer the same tree, that have been previously transmitted and are waiting for the corresponding acknowledge can be "canceled", in the sense that those previous messages are transmitting old state. Since a newer state is being transmitted, to all neighbors (multicast), messages with older state no longer are required to be monitored. This difference between old and newer state is given by the SN.

In the case of a unicast transmission (Interest or NoInterest), referring a given (S,G) tree and destined to a given neighbor, all previous unicast messages referring the same tree and same destination neighbor can be "canceled", in the sense that newer state is being transmitted to the same destination neighbor. This can happen for example when the router changes interest and the previous interest message has not been acknowledged yet.

If there are being reliably transmitted multicast and unicast messages concurrently, being the unicast messages transmitted after the multicast message, an acknowledge to the unicast message also repre-

sents an acknowledge to the multicast message. This can only happen when IamNoLongerUpstream is being transmitted concurrently to Interest or NoInterest messages. Since the interest message has been transmitted after the IamNoLongerUpstream and since both messages indicate that a router is NOT UPSTREAM regarding a given tree, there is no loss of information if the neighbor does not hear the IamNoLongerUpstream message. Also, since the interest message would have a SN greater than the other message, if the network changes the order of these messages, the neighbor would never acknowledge the IamNoLongerUpstream message, since it has a SN lower than the last received message regarding this tree (the interest message).

Also, in case of a message being transmitted concurrently to a neighbor that is synchronizing, if the router's SnapshotSN transmitted to the neighbor router is greater than the SN of the transmitted message, the neighbor router will never acknowledge this message, since fresher state was included in the synchronization process via Sync messages. For this reason, those neighbors are considered to have implicitly acknowledged.

The ACK of a given message must be unicasted to its source and it must include the identification of the tree, the corresponding SN, the BootTime of the source and destination routers and also the MySnapshotSN and NeighborSnapshotSN of the synchronization process. The ACK must only be accepted if these fields are correct.

14 Removal of neighbor's state regarding a tree

A router must maintain all information of its neighbors regarding upstream and interest. This information is obtained by the reception of control messages such as IamUpstream, IamNoLongerUpstream, Interest, NoInterest and Sync messages.

When the interface of a router transitions from root to non-root or vice-versa, and the tree is in a ACTIVE or UNSURE state, it is safe to remove information regarding the interest state stored on those interfaces that was received until that happened. Information regarding neighbors in UPSTREAM state must never be lost and must be transitioned to the new interface type.

Regarding trees in an INACTIVE state, that have been previously in an ACTIVE or UNSURE state, all information of neighbors regarding those trees can be safely removed, since the router no longer is logically connected to the tree.

Regarding trees that transition to UNSURE state, interest information can be safely removed since neighbors no longer consider this router as UPSTREAM and interest information will no longer be transmitted to it. Information regarding UPSTREAM neighbors must never be lost.

Interest information received in a root interface can be safely ignored, since the router is not responsible for forwarding data packets. If this interface then changes its role, i.e. becomes a non-root type, the transmission of an IamUpstream message would cause its neighbors to retransmit interest messages to it, if it is the UPSTREAM neighbor that offers the best RPC.

Sequence numbers must not be removed even when the tree transitions to INACTIVE state. This is to prevent the recreation of the tree if an IamUpstream is replayed or received out of order. The next subsection suggests a mechanism to free resources that are being wasted in sequence numbers. Only that mechanism should be used to remove those sequence numbers.

15 Periodic removal of sequence numbers (CheckpointSN)

The CheckpointSN allows to periodically remove stored SNs at neighbors. This mechanism is optional and only allows to free resources at neighbor routers.

The CheckpointSN is included in transmitted Hello messages. These can be calculated and/or inserted in Hello messages at a periodicity greater than the one used to transmit Hello messages, i.e. include the CheckpointSN from N to N messages.

The CheckpointSN that a router should inform its neighbors must be:

- If control messages are being currently transmitted, the CheckpointSN must be the lowest SN of a control message that is currently being transmitted (and not yet acknowledged), minus one;
- If there are no messages being currently transmitted, the CheckpointSN must be the SN of the last transmitted message (stored at InterfaceSN).

The reason for decrementing by one the lowest sequence number that is currently being transmitted in a control message is to not cause a deadlock. If a message is currently being transmitted, and it was not acknowledged by all routers yet, this means that those routers still require to process a message with that sequence number, for that reason the CheckpointSN must be lower than the SN that is currently being transmitted in an ongoing message.

If the CheckpointSN is intended to be used, a router must store the CheckpointSN of each of its neighbors. This is received in Hello messages. A router by receiving the CheckpointSN from neighbor N:

- If the received CheckpoinSN is greater than the stored one (for N), then the stored CheckpointSN is updated and all stored SNs, about N, that are lower than it are removed.

16 Timers

HelloTimer (HT): controls the periodicity of Hello messages. Default value = 30 seconds.

NeighborLivenessTimer (NLT): Controls the liveness of a neighbor. In case of an ongoing synchronization, the default time is set to 10 seconds (neighbor in a state different than SYNCED). In case the neighbor has finished the synchronization successfully, this timer is set according to the received Hello Hold Time present in Sync or Hello messages (in case this option is not present, set to 105).

SyncRetransmissionTimer: Controls the retransmission of Sync messages in case the neighbor does not answer in due time. Default value = 3 seconds.

RetransmissionTimer: Controls the retransmission of non-Sync messages in case the neighbor(s) does not acknowledge in due time. Default value = 3 seconds.

SourceActiveTimer (SAT): Controls the liveness of an (S,G) source in originator routers. Default value = 210 seconds.

17 Control Message Format

The format of the control messages is similar to the ones used by PIM-DM and PIM-SM, but with some additional fields. Since the protocol is not a standard yet, we do not have any IP protocol number. For test purposes, we used the one specified for PIM (103). Regarding the transmission of multicast messages we use the destination multicast IP address of PIM, i.e. 224.0.0.13.

17.1 Protocol Header

All control messages include a common header:

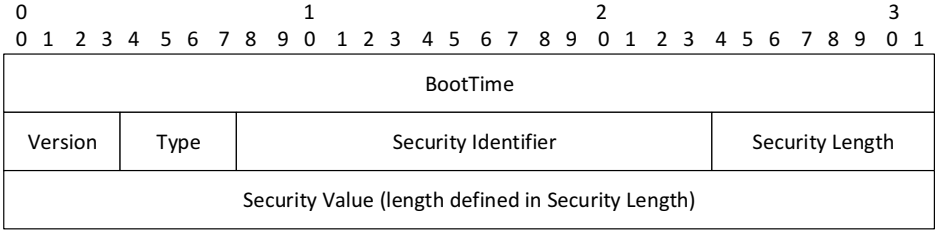

Figure 2: Protocol Header format.

BootTime - Sequence number of the router that is transmitting the message, used to detect reboots and SN overflows;

Version - Version of the protocol implementation (currently set to 0);

Type - Defines the type of the transmitted control message. The following types are available:

- 1 - Hello
- 2 - Sync
- 3 - IamUpstream

- 4 - IamNoLongerUpstream
- 5 - Interest
- 6 - NoInterest
- 7 - ACK

Security Identifier - Identifies the key and algorithm used in the integrity and authentication check (same as KeyID of OSPF messages [1]). In case the Identifier is 0, the message does not offer any integrity nor authenticity guarantees.

Security Length - Number of bytes of the following field. In case this is set to 0, the message does not offer integrity nor authenticity guarantees.

Security Value - Field that includes the cryptographic hash, used to verify the message integrity and authenticity. This value must be calculated in the following way: SecurityValue = function_cryptographic_hash(SourceIP + DestinationIP + Message with Security Value zeroed, Key), with:

- function_cryptographic_hash - function used to generate a cryptographic hash, having as input the secret key (Key) and the content of the message;
- SourceIP - source IP of the IP header;
- DestinationIP - destination IP of the IP header;

17.2 Hello

Hello message can include a variable number of options, since not all options are mandatory.

This message must be multicasted to all neighbors. Hello messages are composed by several options described in a TLV (Type-Length-Value) format.

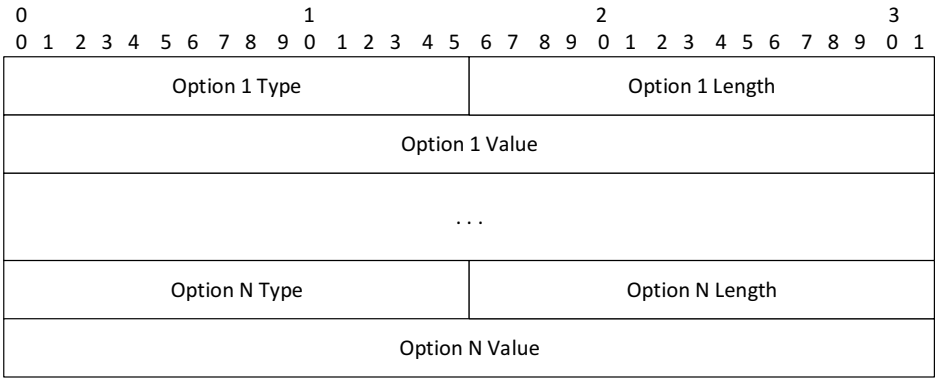

Figure 3: Hello message format.

Option Type - Identifies the option given in the Option Value field. Defined types are the following:

- 1 - Hello Hold Time
- 2 - CheckpointSN

Option Length - Number of bytes that are used by the Option Value.

Option Value - Value of a given option.

Hello Hold Time

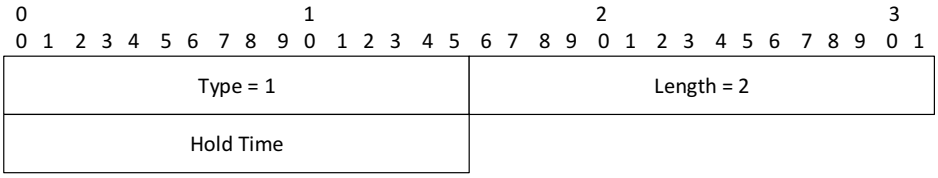

Figure 4: Hello Hold Time TLV.

Hold Time is the number of seconds a receiver must wait until a neighbor is declared dead. If the Hold Time is set to '0', the information is timed out immediately (to accelerate neighbor removals). The Hello Hold Time option is mandatory, so it must be included in Hello messages.

CheckpointSN

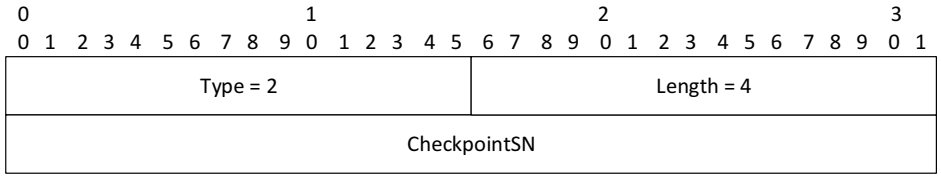

Figure 5: CheckpointSN TLV.

Option used for a router to advertise its CheckpointSN to its neighbors. This option is not mandatory. It corresponds to an optimization used to avoid storing excessive SNs.

17.3 Sync

This message must be unicasted to a single neighbor (the one that is performing the synchronization with the router).

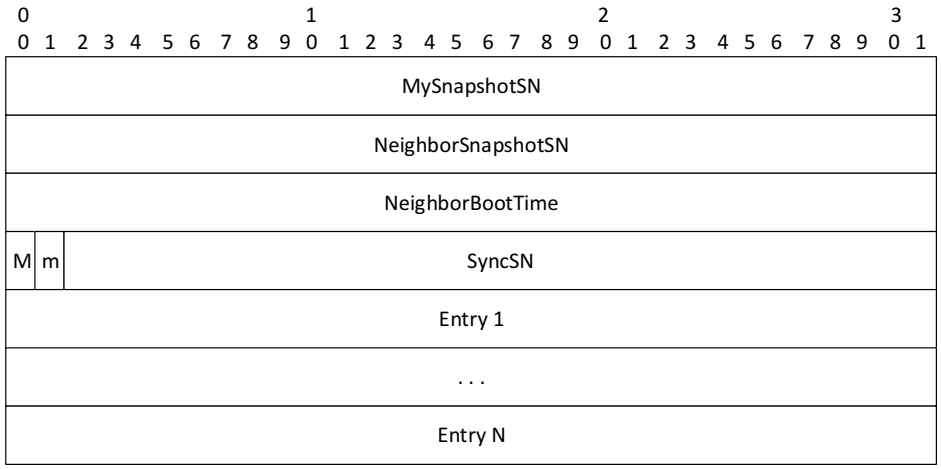

Figure 6: Sync message format.

MySnapshotSN - SnapshotSN of the router transmitting this message;
NeighborSnapshotSN - SnapshotSN of the neighboring router;
NeighborBootTime - BootTime of the neighboring router;
M - Master flag. Set to 1 if the router considers itself Master (neighbor is in SLAVE state);
m - More flag. Set to 1 if the router that is transmitting this message still has unacknowledged tree information;
SyncSN - Sequence number used to fragment Sync messages.
In case the More flag (m) is cleared, the Entry fields carry Hello options, such as the Hello Hold Time and CheckpointSN.
In case the More flag (m) is set, the Entry fields carry information on the trees for which the router considers itself as UPSTREAM (information from the snapshot). These fields must respect the structure below (Sync Tree Entry).

Sync Tree Entry

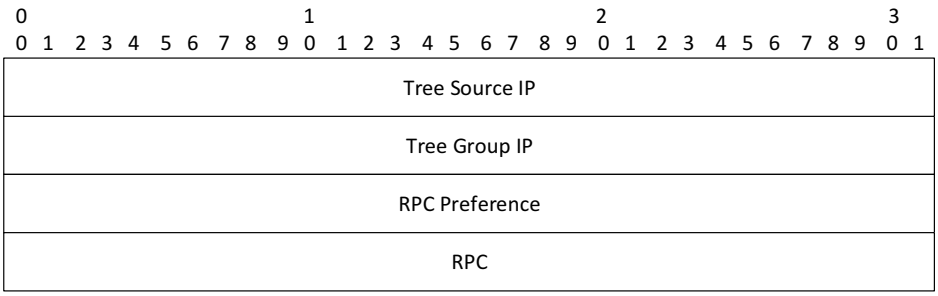

Figure 7: Sync tree entry message format.

Tree Source IP - IP address of the source of multicast traffic;
Tree Group IP - IP address of the multicast group;
RPC Preference - The preference value assigned to the unicast routing protocol that provided the route to the source;
RPC - The RPC to the source. The metric is in units applicable to the unicast routing protocol used.

17.4 IamUpstream

This message must be multicasted to all neighbors.

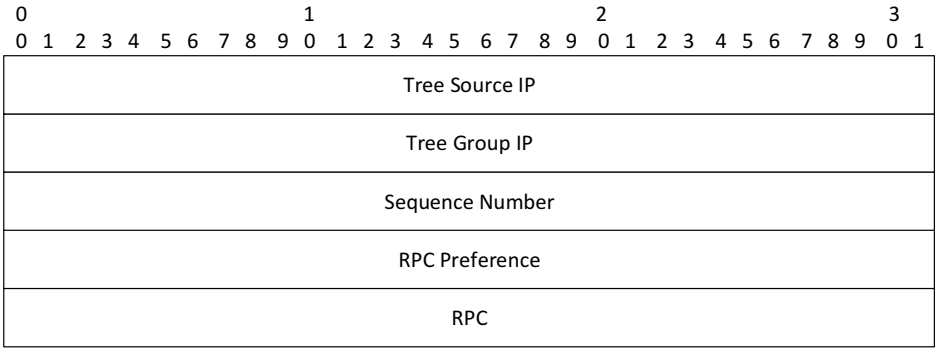

Figure 8: IamUpstream message format.

Tree Source IP - IP address of the source of multicast traffic;
Tree Group IP - IP address of the multicast group;
Sequence Number - Used to sequence messages;
RPC Preference - The preference value assigned to the unicast routing protocol that provided the route to the source;
RPC - The RPC to the source. The metric is in units applicable to the unicast routing protocol used.

17.5 IamNoLongerUpstream

This message must be multicasted to all neighbors.

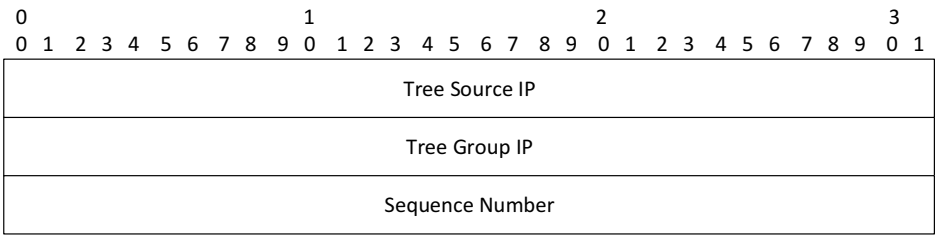

Figure 9: IamNoLongerUpstream message format.

Tree Source IP - IP address of the source of multicast traffic;
Tree Group IP - IP address of the multicast group;
Sequence Number - Used to sequence messages.

17.6 Interest

The format of this message is equal to IamNoLongerUpstream messages. These messages are distinguishable by the message's type at the Protocol Header.

This message should be unicasted to a single neighbor (the AW).

17.7 NoInterest

The format of this message is equal to IamNoLongerUpstream messages. These messages are distinguishable by the message's type at the Protocol Header.

This message should be unicasted to a single neighbor (the AW).

17.8 ACK

This message acknowledges the reception of IamUpstream, IamNoLongerUpstream, Interest or NoInterest messages. This message must be unicasted to the source of those messages.

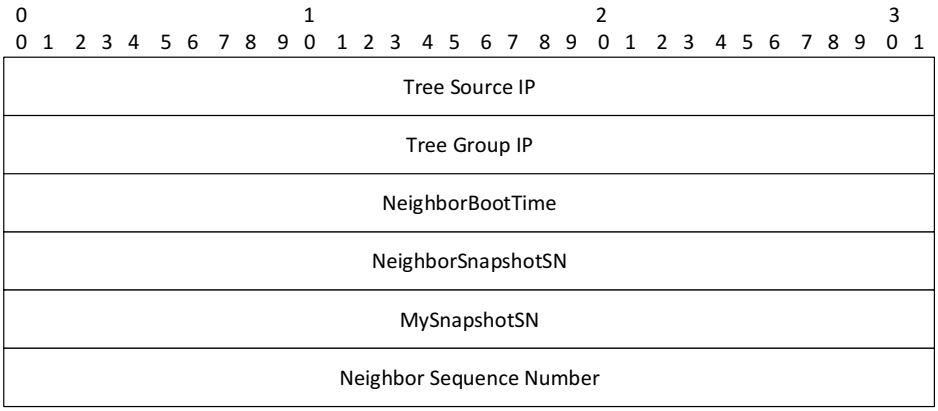

Figure 10: ACK message format.

Tree Source IP - IP address of the source of multicast traffic that this message is acknowledging;

Tree Group IP - IP address of the multicast group that this message is acknowledging;

NeighborBootTime - BootTime of the neighboring router (the one that will receive the ACK message);

NeighborSnapshotSN - SnapshotSN of the neighboring router (the one that will receive the ACK message);

MySnapshotSN - SnapshotSN of the own router (that is transmitting the ACK message);

Neighbor Sequence Number - Sequence number of the message that the ACK is acknowledging;

References

[1] John Moy. Ospf version 2. STD 54, RFC Editor, April 1998.