Hard-state Protocol Independent Multicast – Source Specific Multicast (HPIM-SSM)

Margarida Marques Simões
margarida.simoes@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

January 2021

Abstract

In IP communications, multicast routing protocols provide an efficient way of distributing traffic from a node to a group of nodes in the network. This is achieved through a distribution system usually formed by one or more trees connecting the sources to the receivers through optimal paths. The main multicast protocols and most used currently are from the PIM family which includes the PIM-DM, PIM-SM, and PIM-SSM protocols. Due to their soft-state nature, PIM protocols suffer from slow convergence caused essentially by their slow reaction to events and changes in the network.

In this MSc Dissertation we developed a hard-state version of PIM-SSM, designated by HPIM-SSM, which overcomes several limitations of the current PIM-SSM protocol. We present the specification of HPIM-SSM. The protocol was implemented in Python and tested in a network emulated environment. The correctness of the specification was verified through model checking techniques using the Promela language and SPIN tool. We compared the convergence time of HPIM-SSM and PIM-SSM by creating and executing convergence time tests in both protocols. For this purpose we also implemented the PIM-SSM protocol in Python. Our results show that HPIM-SSM has much better convergence performance than PIM-SSM at the cost of some added stored information, making it suitable for high speed networks.
Keywords: IP Multicast; Multicast routing protocols; PIM; PIM-SSM

1. Introduction

Traditional IP communications support unicast transmissions where a host sends packets to a single host (one-to-one communications) and broadcast communications where a host sends packets to all hosts connected to a specific subnet. IP multicast communications allow a host to send IP packets to a subset of all hosts within the network. It provides a distribution system, usually formed by one or more trees for delivering multicast traffic from sources to groups of interested hosts. For applications that use group communication, e.g., videoconferencing, Internet TV distribution, stock quotes exchanges, or distance learning, the utilization of an IP multicast protocol is essential for efficient management of the resources of the network [14, 9, 15].

Currently, the main multicast routing protocols of the Internet are PIM-DM [1], PIM-SM [3], and PIM-SSM [3]. Between these three, the most popular among the telecommunication operators and more widely used are the PIM sparse mode protocols: PIM-SM and PIM-SSM [6]. However, all PIM protocols suffer from slow convergence due to the slow reaction to events such as link failures, or cost changes. These issues derive from the soft-state nature of PIM protocols that rely on the periodic transmission of control messages to keep the state of routers updated. In addition, there may be losses of the stored state in the routers, leading to network inconsistencies, unwanted removal of trees, and subsequently tree rebuildings. This can cause excessive network congestion and loss of multicast data, a limiting factor for their deployment in high-speed networks. In previous work, a hard state version of PIM-DM designated by HPIM-DM [2] was developed. It keeps the main characteristics of PIM-DM but has a better convergence time, faster reconfiguration in the presence of network failures or unicast route changes, and more resilience to replay attacks.

In this work, we developed a hard-state version of PIM-SSM, designated by HPIM-SSM. HPIM-SSM has a much faster convergence than PIM-SSM and guarantees that the trees built have optimal paths from the receivers to the source. The tree states are stored in the routers and are only removed or changed when a control message or other specific event in the network happens. The routers do not rely on timers to keep their states up-

dated or to transmit control messages. The transmission of control messages is triggered by specific events and is immediate. HPIM-SSM also overcomes some of the PIM issues, like the lack of reliability and sequencing in the transmission of messages. The HPIM-DM protocol has the same guarantees.

The correctness of the specification was verified through model checking techniques using the Promela language [12] and SPIN tool [13]. The protocol was implemented in Python based on its specification and its implementation was tested using NetKit-NG [5], a network emulated environment. PIM-SSM was also implemented in Python and this implementation was used to perform convergence time tests. These tests were also performed in HPIM-SSM, which allowed us to compare the convergence of both protocols regarding a group of network events.

2. State of Art

In this section, we will describe some of the PIM protocols, namely, PIM-DM, PIM-SM, PIM-SSM and HPIM-DM.

2.1. PIM

The Protocol Independent Multicast (PIM) is not dependent on a specific unicast routing protocol, and it uses the information of the unicast routing table to perform the RPF check and create distribution trees. A router can only have one root interface for any entry in the multicast routing table. PIM routers do not send or receive routing updates, which reduces the overhead significantly in the network compared with other multicast routing protocols like DVMRP. PIM has two modes of operation: sparse mode (PIM-SM and PIM-SSM) and dense mode (PIM-DM).

PIM-DM [1] is meant for dense networks since it assumes that all subnets in the network have hosts interested in receiving multicast traffic. It uses source trees to distribute multicast traffic to the hosts, and these trees are built using a flood-and-prune behavior.

PIM-SM and PIM-SSM are meant for sparse networks and work with a completely different strategy of PIM-DM. Trees are built from the bottom to the top and is the interest of the hosts that triggers this process. Routers explicitly request to receive multicast traffic using PIM Join messages that are sent towards the root of the tree. In PIM-SM, a shared tree is used to distribute multicast traffic to the receivers. The root of a shared tree is the RP. There is a shared tree for each active multicast group in the network. Sources must register with the RP to notify it that they are active through a source registration process. Then they can start transmitting multicast traffic to the RP that forwards it downwards the shared tree. PIM-SM enables a last-hop router to switch from the shared to the source tree for a specific source through the source tree switchover process. For each pair (S, G), a different source tree is built having as root the source.

2.2. PIM-SM and PIM-SSM issues

Lack of protection in control messages One of the problems in PIM is the absence of protection in some control messages. The routers do not know if the messages they sent are delivered to the intended receivers, given that messages can be lost in the network. We present now an example showing the possible consequences of the lack of protection of Join messages in the Prune Override Mechanism. PIM routers expect to receive overriding Joins from downstream neighbors that wish to continue receiving traffic in response to a Prune sent by other neighbors in the link. If overriding Join messages are lost or delayed in the network, and the AW does not receive these messages before the Prune override timer times out, it goes to the NoInfo state and the traffic ceases to be transmitted on the link while one or more routers are still interested in receiving it.

Lack of message ordering guarantee Besides the lack of protection in control messages, there is also no guarantee that the transmission order is preserved. If a downstream router sends a Prune to the AW and then sends a Join, but the Prune is the last one to arrive, the AW is pruned, and the router stops receiving traffic despite still being interested. If the opposite occurs, the router sends first a Join and then a Prune, but the messages are received in the inverse order, the AW will keep forwarding traffic with no downstream interested routers.

Slow Tree Reconfiguration When an event that causes a tree reconfiguration occurs, e.g. the RPC changing in a router, the reconfiguration of the tree may only happen through the periodic reconstruction of the tree (which can take 3 minutes) or through the reception of periodic messages (around 1 minute). During this time, the tree may not have the best configuration making multicast traffic not being forwarded through the optimal paths between routers and the source.

Designated Routers on shared links The DR in a shared link is the router responsible for sending Prune and Joins messages on behalf of hosts. The DR is the router with the highest IP address at the link and not the one with the shortest path to the root node. If a host in a shared link joins a multicast group, a path will be installed between the DR of the link and the root node, to deliver the

multicast traffic to the interested host. If the DR is not the router in the link with the shortest path to the source, the path installed will not be the optimal one.

2.3. HPIM-DM

HPIM-DM (Hard-state Protocol Independent Multicast - Dense Mode) is a hard-state version of PIM-DM that was developed on a previous MSc Dissertation [2]. Like PIM-DM, it is meant for dense networks, and it is IP routing protocol-independent. One main difference from PIM-DM is that routers maintain knowledge of the existing multicast trees at all times by keeping information about all upstream routers from which multicast traffic can be received. Another difference is the implementation of a synchronization process that allows a router joining the network to immediately learn which trees are active and thus start receiving multicast traffic if interested.

3. HPIM-SSM Specification

HPIM-SSM (Hard-state Protocol Independent Multicast – Source Specific Multicast) multicast routing protocol is a hard-state version of PIM-SSM meant for sparse networks, that aims to react faster to network changes, and have better resilience to replay attacks and reliability in the transmission of control messages. In section 2.2.4 we presented some issues of the PIM-SM/PIM-SSM protocols that we propose to solve or at least minimize. In the specification, we will discuss the type of control messages, and the states needed to be stored in the routers. Besides the tree maintenance, we will cover the synchronization process and the reliability and sequencing of the messages.

3.1. Protocol Overview

Before entering into the details of the protocol, we present an overview of its main concepts and ideas. As PIM-SSM, HPIM-SSM is an IP multicast routing protocol that relies on a unicast routing protocol to perform the RPF checks. The creation of trees is also triggered by hosts manifesting interest in a specific source and group using the IGMPv3/MLDv2 protocols. With a hard-state version of PIM-SSM, we ensure a faster convergence of the protocol and eliminate the need of periodic transmission of Join messages, the periodic reconstruction of trees, and the delays caused by timers. To built a hard-state version we need to ensure that control messages are received and processed in the correct order in each router. Like PIM-SSM, HPIM-SSM uses the Join/Prune messages to create the source trees and the Assert to elect a single forwarder in each link, which is also responsible for sending the interest upwards. In HPIM-SSM, the Assert is triggered

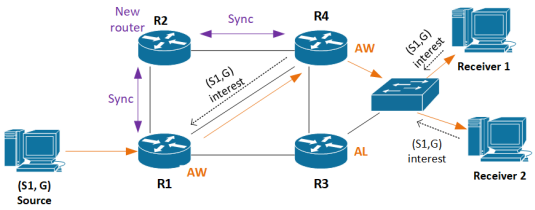

Figure 1: HPIM-SSM Overview

by the arrival of interest at a link and this interest can be of routers or hosts. With this, we avoid the need for the DRs and all the problems that PIM-SM/PIM-SSM suffer from the relation of the DR with the Assert (explained in section 2.2.3.A). An interface can only forward and send interest upwards if it is in the AW state, and in every link, an AW must be elected. In the presence of interest, the non-root interfaces can be AWs and the root interfaces can become non-root, and therefore, potential AWs. Given that, all interfaces need to store the interest and the RPC of their neighbors. Due to the hard-state nature of the protocol, when a router joins the network it is necessary a synchronization process with its neighbors so it can learn information about the existent trees in the network.

Figure 1 illustrates the main concepts of the protocol described above. It shows a network scenario where initially R1, R3, and R4 are running HPIM-SSM and both Receivers are interested in (S1, G) multicast traffic. R4 has a better RPC than R3. When the receivers manifest their interest, R3, and R4 save it, the Assert is triggered, and R4 is elected AW since it has better RPC. R4 sends the interest to its next hop router´in the unicast routing table for the source S1. When R1 receives the interest, saves it, and becomes AW. It does not send the interest upwards since it is directly connected to the source. The orange arrows represent the tree created and the path used to forward the (S1, G) traffic from the source to the receivers. After the tree being created, R2 starts the HPIM-SSM protocol and synchronizes with its neighbors (R4 and R1) to learn information about the active trees in the network. Each purple arrow represents the synchronization process between two neighbors.

In section 3.2 we explain the Hello protocol, in section 3.3 the concepts of creating and maintaining trees, and in section 3.4 and 3.5 the interest and assert state machines respectively. Section 3.6 covers the removal of tree state, section 3.7 the installation and removal of a tree in the network, and section 3.8 the synchronization process and the information that needs to be exchanged

initially. In section 3.9 is covered the message sequencing and reliability mechanisms and in section 3.11 the message's format. Finally, in section 3.11 is covered the existence of loops in the HPIM-SSM protocol.

3.2. Hello Protocol

The Hello protocol in HPIM-SSM is the same as in HPIM-DM and its description can be accessed on the folder "docs" of the GitHub repository [7].

3.3. Tree Formation and Maintenance Concepts

Two types of messages are used in the tree formation and maintenance procedure: (i) Assert messages used in all links to elect the AW and (ii) Join/Prune messages to inform neighbors about the interest of a router in joining or leaving an (S, G) source tree. Assert, Join, and Prune messages are multicasted between neighbors (link-local scope). When a router receives a notification of interest on a non-root interface and the interface becomes AW, it propagates the interest towards the source of the tree. A router will indefinitely consider a downstream neighbor being interested until it advertises otherwise. A tree is removed when there are no interested receivers left for the corresponding source by the transmission of Prune messages towards the source.

3.4. HPIM-SSM Interest

The interest/no interest of the hosts triggers the creation/removal of the trees. Interfaces know if there are any hosts interested with the IGMPv3/MLDv2 protocol. The creation/removal of trees is done by sending Join/Prune messages upwards until reaching a router directly connected to the source. The Join message indicates to the receiving router that the sending router is interested in receiving multicast traffic from a specific (S, G) tree, and the Prune message indicates the opposite. When an interface receives a Join (S, G) message from a neighbor, it stores the interest of the neighbor in receiving multicast traffic from the (S, G) tree. This information is only removed upon the reception of a Prune (S, G) message from the same neighbor. Routers send Join/Prune messages when there is a change in their interest state. Both Join and Prune messages are multicasted through root interfaces and include the IP address of the source (S) and the multicast group (G).

Root interfaces receive multicast traffic according to the interest state of the router, and non-root interfaces transmit the multicast traffic according to their forwarding state. The interest state of a router depends on the forwarding state of its non-root interfaces. The forwarding state of a non-root interface depends on its Assert state. The Assert states will be detailed in section 3.2.5. For now, we will only mention the AW state, already covered in the description of the other PIM protocols. The definition of these states is:

• A non-root interface can be in two states regarding the downstream interest on an (S, G) tree: DOWNSTREAM INTERESTED (DI) or NOT DOWNSTREAM INTERESTED (NDI). It is in the DOWNSTREAM INTERESTED state if it is connected to at least one interested neighbor or host and in the NOT DOWNSTREAM INTERESTED state otherwise.

• A non-root interface can be in two states regarding the forwarding state on an (S, G) tree: FORWARDING or PRUNED. The forwarding state of a non-root interface depends on its Assert state. It is FORWARDING if it is the AW and is PRUNED otherwise.

• A router can be in two states regarding the interest on an (S, G) tree: INTERESTED (I) or NOT INTERESTED (NI). A router is INTERESTED if it has at least one non-root interface in the FORWARDING state and is NOT INTERESTED otherwise.

3.5. HPIM-SSM Assert

The Assert is used in point-to-point and shared links to elect an AW. The AW is the router responsible for forwarding the traffic to the link as well as propagating the interest towards the source. Without the election of the AW, if a downstream router sent a Join message to a shared link, every router would receive it and propagate this interest towards the root of the tree, creating parallel paths and duplication of traffic.

The AW is the non-root interface connected to the link that has the lowest RPC value to the source (in case of a tie, the interface with the highest IP address wins), thereby assuring the creation of an optimal path between the hosts and the source. The Assert message is used to elect the AW of a link for a specific source and group and contains the RPC of the sending router to the multicast source. It is multicasted through non-root interfaces and includes the IP address of the source and multicast group. The Assert process developed has the following properties:

• It has 2 states: Assert Winner(AW) and Assert Loser(AL)

• Only the non-root interfaces have Assert state regarding an (S, G) tree

• A non-root interface in the DOWNSTREAM INTERESTED state can be in the AW or AL state

• A non-root interface in the NOT DOWNSTREAM INTERESTED state is always in the AL

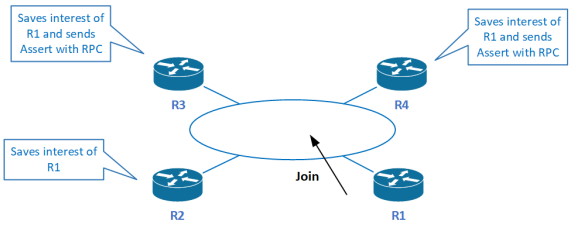

Figure 2: Interest and Assert

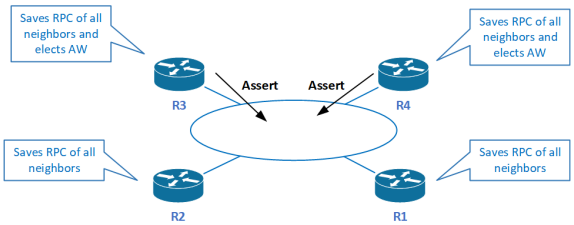

Figure 3: Assert process

state

Relation Interest-Assert and Stored States

The AW must have updated interest information to decide if it can remain AW and act as the link forwarder. The interest information sent by a neighbor must always be stored by the receiving interface irrespective of the current interface type and state (root, non-root, AW, AL) since any interface can become an AW. The AW election is triggered by the arrival of a Join message or interested hosts. Therefore, when the AW is elected it already has interest information. There is then a coupling between the interest and assert states.

Initially, all non-root interfaces in the link in the DI state consider themselves being the AW and transmit an Assert message containing the RPC of their routers. The root interfaces can not have Assert state and so, do not send an Assert message. If a non-root interface receives interest for some tree (S, G) but has no route on its unicast routing table for the source S it sends an Assert message with infinite cost. All interfaces in the link (root and non-root) receive the Assert messages and save the RPC contained in them. The non-root interfaces will then use the RPCs stored to elect an AW. The root interfaces do not have an Assert state, but save the RPCs because they can become non-root interfaces and, therefore, potential AWs.

Figures 2 and 3 show the trigger of the Assert and the Assert messages exchanged when R1 multicasts a Join message to the shared link. In both figures, R1 and R2 are connected to the link through a root interface and R3 and R4 through a non-root interface. All interfaces save the interest of R1 and the non-root interfaces of R3 and R4 send an Assert message with their RPC. All interfaces save the RPCs included in the Assert messages and the non-root interfaces of R3 and R4 elect an AW.

If there is a change of the RPC (without interface role change) in a non-root interface in the DI state (AW or AL), that interface has to send an Assert with its new RPC. Even if another router is elected AW, no additional messages need to be transmitted because interfaces know the more updated RPC of each other. If at a given moment there are no interested neighbors or hosts left for a specific (S, G) tree, the non-root interfaces will change from the DI to the NDI state and send an Assert Cancel message. An Assert Cancel consists of an Assert message but instead of containing the RPC of the router, it contains a predefined value (very high value), that acts as an infinite metric. It is used to alert the neighbors, the router can no longer be the one responsible for forwarding data in the link and its RPC can be removed. When an interface receives the Assert Cancel, it removes the RPC of the neighbor that sent the message. The other events that trigger the transmission of Assert Cancel messages are when an interface in the DI state (in the AW or AL state) becomes root or is removed.

Initially, it was considered in our specification and implementation of HPIM-SSM an Assert similar to the one of PIM-SM/PIM-SSM protocols. This Assert had the same four properties mentioned above but routers only stored who the AW was and its RPC. The number of Assert messages needed to be transmitted was higher. We then identified some scenarios where different routers disagreed on who the AW was, although only one router considered itself the AW. We then developed a new Assert, the one we just described, that shows to be more efficient (less transmission of messages) and correct and with a simpler state machine.

3.6. Removal of tree state

The removal of the tree state is important, otherwise, routers would maintain indefinitely their multicast routing tables referring to (S, G) trees that do not exist anymore in the network. This state would only be removed by restarting the HPIM-SSM protocol in the routers. Removing the tree state is removing the entry in the multicast routing table regarding the (S, G) tree, including all downstream and the upstream interfaces associated with it. A router can remove the state regarding an (S, G) tree when all downstream interfaces are in the NDI state regarding that tree, and no assert information is stored. An interface has no assert information when no RPCs are stored of any neighbor. An interface removes the RPC of a neighbor when it

receives an Assert Cancel message from it, when the neighbor fails, or when its Boot Time changes.

3.7. Installing and removing a tree

When a non-root interface receives the interest of a host or router referring to a new tree, it becomes DOWNSTREAM INTERESTED for that (S, G) tree. This will trigger the Assert, and if the interface becomes the AW, it changes to the FORWARDING state. The router then changes to the INTERESTED state since it has a non-root interface in the FORWARDING state and sends a Join (S, G) message through its root interface. The Join message will be received by the upstream routers, and the process is repeated until it reaches the first-hop routers. When a first-hop router is in the INTERESTED state, and the source is active, traffic is forwarded down the tree by the routers in the INTERESTED state until it reaches the interested hosts. Tree removal or pruning follows the same logic. When all non-root interfaces of the last-hop routers do not have any more hosts interested in receiving multicast traffic for a specific tree, they become NOT DOWNSTREAM INTERESTED, change to the NI state, and consequently to the PRUNED state. The routers then change to the NOT INTERESTED state since they no longer have non-root interfaces in the FORWARDING state and send a Prune (S, G) message through their root interfaces. The Prune message will be received by the upstream routers, and the process is repeated until it reaches the first-hop routers. When all routers are in the NOT INTERESTED state, it means that no hosts in the network are interested in receiving the multicast traffic for that source and the traffic is no longer forwarded.

3.8. Synchronization

Similarly to HPIM-DM, during the synchronization process, a new router learns information from its neighbors that allow it to integrate in the network correctly. In our case, since Join messages are not sent periodically, and the formation of the tree, as well as the Assert, is triggered by them, the new router must receive the equivalent of the information contained in the Join messages previously sent in the link, i.e. learn for which trees the routers connected to the link through a root interface (downstream routers of the link) are in the INTERESTED state. The new router must also learn the RPC of the routers that are connected to the link through a non-root interface in the DOWNSTREAM INTERESTED state since these routers sent an Assert with their RPC previously to the link. All routers in the link are already storing these RPCs to later perform an AW election locally when triggered. Remember that the local AW election does not trigger the transmission of Asserts even for the router that became AW.

The process that assures the consistency of synchronization and the synchronization protocol in HPIM-SSM is the same as in HPIM-DM and its description can be accessed on the folder ”docs” of the GitHub repository [7].

3.9. Message Sequencing and Message Reliability

The message sequencing and message reliability mechanisms are similar to the ones used in HPIM-DM and its description can be accessed on the folder ”docs” of the GitHub repository [7].

3.10. Message Format

We present below the format of the protocol header that all control messages include as well as the specific fields of each one of the control messages. The format of all the control messages (including this protocol header) used in our implementation is identical to the ones used in HPIM-DM and can be accessed on the GitHub Repository [7] in the folder “docs/HPIMStateMachines.pdf”.

Protocol Header The packet of all control messages includes this header that contains the following fields: BootTime, Version, Type, Security Identifier, Security Length, and Security Value. The definition of all fields except the Type is equal to the HPIM-DM protocol. The field Type determines the control message that is being transmitted and can have the following values: 1-Hello; 2-Sync; 3-Assert; 4-Join; 5-Prune; 6-Ack.

Hello, Assert, Join/Prune, ACK The Hello and ACK message’s format is equal to the ones of HPIM-DM. The format of the Assert message including the definition of the fields is equal to the IamUpstream message of HPIM-DM. The Join and Prune messages have the same format in HPIM-SSM, having as the only difference the message’s type at the Protocol Header. Besides, this format is equal to the Interest messages’ format of HPIM-DM.

Sync The Sync messages format is similar to the one of HPIM-DM. All fields are the same except the Sync Entry field. In HPIM-SSM, the Sync Entry field can have two different formats, depending on the information being exchanged. During the synchronization process, interest and assert information can be exchanged. To distinguish between these two types of information, every Sync Entry field includes an Identifier, that can have the value 0 or 1 if interest or assert is being exchanged respectively. When interest is being exchanged, the Sync Entry field has the following subfields: the Identifier, the Tree Source IP, and the Tree Group IP. When assert is being exchanged, the Sync Entry field has the following subfields: the Identifier, the Tree Source IP, the Tree Group IP, the RPC, and the RPC Preference. The RPC Preference is

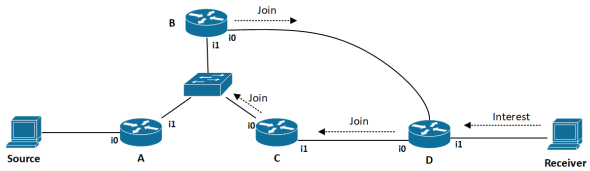

Figure 4: HPIM-SSM Loop Network

the preference value associated with the unicast protocol that provides the route to the source and the RPC is the cost to the source. For both cases, the Tree Source IP and Tree Group IP represent the IP of the source and IP of the multicast group of a certain tree (S, G).

Initially, it was considered another format for the Sync Entry field of the Sync messages that showed to be not correct in a specific scenario. In our implementation, the Sync Entry field has not been updated yet to the new format.

3.11. Existence of loops in HPIM-SSM

It can happen in a network with routers running HPIM-SSM, the creation of a loop. By loop, we mean that routers store tree state indefinitely, which can only be removed by restarting the protocol. A loop can only happen when the routers are running different unicast protocols. Figure 4 is an example of a network scenario that would create a loop. Every router except router B is running OSPF as the unicast routing protocol. Router B has a static route configured to D. Each router has a root and a non-root interface. The interfaces i0 are root and the interfaces i1 are non-root. Router D is notified that the receiver is interested in receiving multicast traffic from a certain (S, G) tree and sends a Join message through its root interface. Router C does the same and the Join is multicasted in the shared link triggering the Assert. Because router B has the static route, it wins the Assert. It sends a Join through its root interface (to router D), creating a loop. Router D does not send a Join message again because it was already in the INTERESTED state for that tree. If the source starts transmitting, the traffic will not reach the receiver since router A is not part of the tree.

This loop is a consequence of the obligation of an Assert process to decide who forwards traffic to the link and send Joins upwards. In fact, a router can be elected AW without being the next-hop router of any downstream neighbor in the link. In this example, the next-hop router of router C is router A and not router B.

4. HPIM-SSM Correctness Tests

We developed a model in the Promela language that was tested with the SPIN tool [13] to verify the correctness of the HPIM-SSM protocol specification. More precisely, we wanted to prove that the state machines of the HPIM-SSM protocol are correct, so that regardless of the events that occur in the network and their order, the routers always converge to the correct state.

We modeled the assert and interest state machines in a single model to verify the correctness of each one as well as the combination of the two. The two state machines are highly linked, since the interest triggers the assert and the forwarding state of an interface depends on the assert and interest states. We did not model the synchronization process since it was already tested in a previous work (correctness of the HPIM-DM protocol [2]) and the process is the same in both protocols, having as only difference the type of information exchanged in the Sync messages.

The code developed and used in this verification can be accessed in our GitHub repository [10] in the branch “Promela”.

5. Protocol Implementations

In this section, we address the implementation of the HPIM-SSM and PIM-SSM protocols. Both protocols were implemented in Python3 and made to run in systems with the Linux operating system. Using Linux has several advantages, namely, the possibility of changing the multicast routing table and define the root and non-root interfaces for each multicast tree. We chose Python because it is a high-level language that allows us to abstract from the machine language. It has available high-level functions that implement multicast and networking operations that in other languages it would be much harder to implement and possibly with several limitations. Python has good readability and offers good documentation. It is an object-oriented language and that allowed us to build an implementation based on classes. Each class includes a set of methods and variables that perform very well-defined functions in the operation of the protocol. Classes are connected by association or inheritance.

The implementation of the HPIM-SSM and PIM-SSM protocols was based on the HPIM-DM [7] and PIM-DM [8] implementations respectively.The PIM-DM implementation is part of the previous work that developed the HPIM-DM protocol. The structure of the protocols, including the classes and their methods, is equivalent to the one of PIM-DM and HPIM-DM protocols and the state machines were adapted to the new protocols.

HPIM-SSM

The implementation of the HPIM-SSM protocol was made according to its specification (section 3) and can be accessed in our GitHub repository [10]. It has the goal to demonstrate that the specifica-

tion, our principle, is feasible and can function in real environments. Moreover, it would be not possible to perform the convergence time tests, that contribute to the convergence time analysis of the protocols, without an implementation. The IGMPv3 protocol was not implemented. Although the hosts must manifest their interest, all the other protocol functionalities could be implemented and tested, not being dependent on the IGMPv3.

PIM-SSM

The implementation of the PIM-SSM protocol was made according to the RFC7761 [3] and can be accessed in our GitHub repository [11]. The implementation was needed to perform the convergence time tests, and compare the times obtained with the ones of the HPIM-SSM protocol.

Due to the short time, we do not implement the designated router functions mentioned in the RFC7761. For the tests we performed and the events to which the time analysis was made, the designated routers' operations were not necessary to be implemented.

6. HPIM-SSM implementation tests

To verify that the protocol was implemented correctly we developed some tests in Python. We used the NetKit-NG software to perform the tests since it emulates a network environment and creates virtual machines. The routers need to run a unicast routing protocol. We chose the OSPF protocol which is being executed with Quagga[4].

All these tests were performed with the implementation of the protocol having the old assert. The synchronization process and its mechanisms are the same but the information being exchanged in the Sync messages was different (only interest information was exchanged). The final assert and the update of the information being exchanged in Sync messages were developed after and no extensive implementation tests were made due to the short time. Nevertheless, the correctness of the assert and its relation with the interest was verified with model checking techniques (section 4). The new Assert implementation, as well as the PIM-SSM implementation, were tested and corrected during the development of the convergence time tests.

Apart from the network with the routers running the HPIM-SSM protocol we created a management network consisting of a central node (a router) connected to all other routers through point-to-point links (each one in a different subnet). This central node received logs from all routers containing the transmission and reception of control messages, as well as the state changes (assert, interest, neighborhood). All these events were registered in the logs with an associated time. To ensure all routers were synchronized we configured the NTP (Network Time Protocol) with the central node being the server and all other nodes the clients. We tested the synchronization with and without trees, the creation and removal of trees, and the forwarding of multicast data with an active source. Each one of this group of tests was performed in an automated way.

Event	PIM-SSM	HPIM-SSM
Pruning the tree when last interested neighbor becomes not interested	= 3 sec. (without Prune loss)	Immediate
	[150, 210] sec. (with Prune loss)	≈ 2 sec
Router gains interest with Join loss	= 60 sec.	≈ 2 sec
Neighbor becomes not interested, but router is still interested	≤ 2.5 sec. (without Join loss)	Immediate
	[3, 5.5] sec. (with Join loss)	
Router joins the network	≤ 180 sec.	Immediate
AW loses its role due to RPC increase	≤ 180 sec.	Immediate
AW loses its role by becoming root and Assert Cancel is lost	≤ 180 sec.	≈ 2 sec
Join/Prune arrive out of order	≤ 210 sec. (if Join is last)	Immediate
	≤ 60 sec. (if Prune is last)	

Figure 5: Convergence time values

7. Protocols convergence time and stored state

7.1. Convergence time

In this section, we compare the convergence time of the HPIM-SSM and PIM-SSM protocols. There were identified some events where the performance of the protocols is significantly different. We study the behavior of the protocols regarding each of the events by comparing the time they took to converge to the correct state. We address the theoretical convergence time values of both protocols regarding the events defined and we discuss the convergence time tests performed in protocol implementations to obtain practical values for the convergence time.

Convergence time: theoretical values

For each event, we calculated the theoretical upper limits of the time the protocols would take to converge to the correct final states. This information is summarized in Figure 5. The upper limits were calculated based on the RFC7761 [3] for PIM-SSM and based on our specification (section 3) for HPIM-SSM. We can easily observe that for HPIM-SSM all these events have almost an immediate reaction while in PIM-SSM values as high as 210 seconds can be reached. Although in the HPIM-SSM protocol messages can also be lost, the reliability of message transmissions ensures very fast recovery and convergence of the protocol. In the events with message loss, for the theoretical and practical values, we assumed only the first message was lost.

Convergence time: practical values

For the first three events of last section, we reinforced our confidence in the theoretical values obtained by performing convergence time tests.

We did not perform these tests for the rest of the events due to their implementation complexity and not having the IGMPv3 implementation

We used the implementations of the HPIM-SSM and PIM-SSM protocols to perform the tests. These tests simulate the events previously described, by creating an initial scenario in the network, triggering the event, and then measuring the time for the routers to converge to the correct final states. All the tests implemented can be accessed in our GitHub repositories: [11] for PIM-SSM and [10] for HPIM-SSM.

Similarly to the implementation tests of HPIM-SSM described in section 6, we created a management network, with a central node connected to every router by a point-to-point link. The central node received logs from all routers containing all state changes and reception/transmission of messages. All these events were registered in the logs with a time associated. All routers had the NTP protocol configured. They were the NTP clients and the central node the NTP server. For the convergence time tests, is very important that all routers are synchronized in time, and with the NTP we assured that. The convergence times were calculated from the logs, by seeing the time difference between the initial and final state of routers. The results obtained were in accordance with the theoretical values presented in Figure 5.

7.2. Stored State

In section 7.1, we compared the time PIM-SSM and HPIM-SSM take to converge and conclude the HPIM-SSM has much faster convergence and reaction to network events. In this section, we compare the amount of information that needs to be stored at the routers for both protocols. It is clear that HPIM-SSM requires more stare information but less timers than PIM-SSM.

HPIM-SSM needs to store more state information regarding each neighbor than PIM-SSM. The sequencing and reliability of messages mechanisms, as well as the synchronization process, need to use several sequence numbers and some timers. All interfaces have to store tree state information regarding the neighbors i.e. assert and interest information. An interface has to store for each neighbor, tree state (its RPC and interest) and the synchronization state (UNKNOWN, MASTER, SLAVE, SYNCED). It also has to store sequence numbers regarding each neighbor that include the BootTime, the neighborSN, the CheckpointSN, the SyncSN, and the SnapshotSN. An interface with at least one neighbor needs to store its BootTime, interfaceSN, Hello Timer, and who its neighbors are. All interfaces need to store their type (root or non-root), and state regarding the ACK-protection mechanism (Retransmission Timer and pending acknowledgments). Regarding a tree (S, G), each interface saves its Assert, downstream interest and forwarding state. It also has to store state regarding the IGMP/MLD protocols to know if any local hosts are interested. Finally, each router needs to save the source and multicast group IP addresses of each existent tree, as well as its RPC and interest state (INTERESTED or NOT INTERESTED).

PIM-SSM does not store so many state information but uses more timers than HPIM-SSM. The Hello and Neighbor Liveness Timers are used by both protocols but PIM-SSM needs to use additional timers to control the Assert, the forwarding state of an interface and the pruning of links. PIM-SSM uses the following timers: Assert Timer, Prune Pending Timer, Join Timer, Override Timer, Expiry Timer, Hello Timer and Neighbor Liveness Timer. Regarding sequence numbers, each interface only needs to store the generation ID of each neighbor. Regarding the interest information, PIM-SSM needs to store the downstream state of non-root interfaces, but does not need to store this interest in root interfaces like it happens in HPIM-SSM. Regarding the Assert information, it needs to be stored in all interfaces who the AW is and the interface Assert state. Contrarily of HPIM-SSM, PIM-SSM only needs to store the RPC of the AW and not of all neighbors. Finally, the upstream state of the router (Joined or not Joined) has to be stored.

8. Conclusions

IP Multicast routing is essential for group communication applications since it helps saving network resources, being much more efficient and adequate than unicast or broadcast. PIM is one of the main multicast routing protocols with the sparse-mode protocols PIM-SM and PIM-SSM being the most popular and widely used. Its deployment is growing, and so, clients expect reduced traffic loss and fast convergence. Improvements made to the current protocol will lead to better performance and better service delivered by IP multicast applications. That is the motivation for our work. Some major limitations of the PIM protocols are slow convergence, slow tree reconfiguration, and network overhead caused by the periodic transmission of control messages and rebuild of the trees. There is also a lack of protection and ordering guarantees in the control messages. More specifically, in PIM-SSM since the trees are built from the bottom to the top and the DR can be the router not having the best RPC to the source, while the Assert is not triggered, the tree will not have the optimal path from the source to the interested hosts. It can happen that the assert is never triggered, or in the

case that is triggered, it can happen a periodic reconstruction of the tree which creates instability in the network.

We developed a hard-state version of PIM-SSM, designated by HPIM-SSM, that has faster convergence and reacts to network changes almost immediately. These network changes are, for example, the appearance of a new router in the network, change in interface costs, change of interface roles, or rebuild of the tree due to change of interest in a router. Trees are built from the bottom to the top with an optimal path from the interested hosts to the source. In HPIM-SSM, the Assert is triggered by the interest and elects an AW in all links, as the interest is propagated towards the source. This ensures the creation of a tree with an optimal path from the hosts to the source and eliminates the need for the DRs. HPIM-SSM ensures that control messages are processed in the order they were transmitted and ensures the messages are received by all routers that should, due to a message sequencing and message reliability mechanism, respectively. Due to its hard-state nature, there is no periodic transmission of messages or periodic reconstruction of the trees. This reduces considerably the number of control messages transmitted in the network and reduces the data packets that could potentially be lost during these reconfigurations, contributing to reduced network overhead.

In this report, we presented a specification of the HPIM-SSM protocol that was then verified through model checking techniques. This verification is important to ensure that regardless of the network events happening in the network, or the order in which the routers in a link receive a certain control message, the assert and interest machines are always respected and that the routers always converge to the correct states. It was also verified the correctness in the interaction between the assert and interest state machines. The protocol was implemented in Python, based on the specification of HPIM-SSM, and tested using a network emulated environment. To compare the convergence time of the HPIM-SSM with PIM-SSM, it was also implemented in Python the PIM-SSM protocol. To evaluate our protocol and compare its convergence with the PIM-SSM protocol, we developed a set of convergence tests that were performed in our implementations of HPIM-SSM and PIM-SSM. The results obtained show that, for the events studied, the HPIM-SSM protocol has much faster convergence than PIM-SSM and the reaction is almost immediate. Also, significant time differences were noticed in the tests with loss of control messages, having HPIM-SSM a much better performance due to the reliable message transmission mechanism.

References

[1] A. Adams, J. Nicholas, and W. Siadak. Protocol independent multicast - dense mode (pim-dm): Protocol specification (revised). RFC 3973, RFC Editor, January 2005.

[2] Pedro Francisco Carmelo de Oliveira. Robust multicast routing protocol. Technical report, October 2018.

[3] B. Fenner, M. Handley, H. Holbrook, I. Kouvelas, R. Parekh, Z. Zhang, and L. Zheng. Protocol independent multicast - sparse mode (pim-sm): Protocol specification (revised). STD 83, RFC Editor, March 2016.

[4] Paul Jakma. Quagga routing suite. https://www.quagga.net/. Last accessed 20 May 2020.

[5] Netkit-NG. Netkit-ng homepage. https://netkit-ng.github.io/. Last accessed 20 December 2019.

[6] Wendell Odom, Jim Geier, and Naren Mehta. *CCIE Routing and Switching Official Exam Certification Guide (Exam Certification Guide)*. Cisco Press, 2006.

[7] P. Oliveira. Hpim-dm implementation. https://github.com/pedrofran12/hpim_dm.git. Last accessed 20 December 2020.

[8] P. Oliveira. Pim-dm implementation. https://github.com/pedrofran12/pim_dm.git. Last accessed 20 December 2020.

[9] Eric Rosenberg. *A Primer of Multicast Routing*. Springer, 2012.

[10] M. Simões. Hpim-sm implementation. https://github.com/Marga97/hpim_sm.git. Last accessed 2 December 2020.

[11] M. Simões. Pim-ssm implementation. https://github.com/Marga97/pim_ssm.git. Last accessed 2 December 2020.

[12] Spinroot. Promela manual pages. http://spinroot.com/spin/Man/promela.html. Last accessed 20 December 2019.

[13] Spinroot. Spin - formal verification. http://spinroot.com/spin/whatispin.html. Last accessed 20 December 2019.

[14] Beau Williamson. *Developing IP multicast networks*, volume 1. Cisco Press, 2000.

[15] Ralph Wittmann and Martina Zitterbart. *Multicast communication: Protocols, programming, & applications*. Elsevier, 2000.