Evaluation of an Overlay for Source-Specific Multicast in Asymmetric Routing environments

Vasco Pereira, Edmundo Monteiro
Department of Informatics Engineering
University of Coimbra
{vasco,edmundo}@dei.uc.pt

Paulo Mendes
NTT DoCoMo Euro-Labs
mendes@docomolab-euro.com

Abstract— **Multicast routing is again gaining popularity with the development of new group services, like real-time multimedia streaming. The aim is to reach big groups of users with the quality level they expect, while using network resources in an efficient way. This is a challenge to be fulfilled as current multicast protocols still have difficulties in dealing with basic issues like asymmetric routing. In this work we describe and evaluate an overlay that allows the use of the source-specific multicast standard in environments with asymmetric routing. A set of tests is made in the data and control plane, to expose the pros and cons of this new proposal.**

Keywords- Multicast routing, routing asymmetry, network simulation

I. INTRODUCTION

The deployment of real-time group communications with efficient use of network resources implies that multicast data should always travel in the best possible path. To accomplish this, routing asymmetries, as well as *Quality of Service* (QoS) requirements, should be considered when building multicast trees. However, these requirements are not accomplished by most IP multicast routing protocols. One of the reasons is that those protocols build multicast trees from the receivers to the sender, while data travels in the reverse direction. The result is that data is forced to travel in a path that is not optimized and sometimes not even suitable. This fact may lead to a loss of performance and to the failure to deliver the quality levels expected by the users. The routing asymmetries that cause this situation are an usual presence in the Internet [1] and can happen due to distinct factors, such as different paths for each direction, same path but different bandwidth for each direction in the same link, as well as quality of service or network access restrictions. Routing policies and traffic engineering may also cause routing asymmetry [2].

In this paper, a new protocol called *Overlay for Source-Specific Multicast in Asymmetric Routing environments* (OSMAR) [3] is described, and evaluated using the *Network Simulator version 2* (NS2) [4]. The aim of the OSMAR protocol is to overcome the lack of adaptation of current multicast protocols to asymmetric routing. Specifically, OSMAR protocol allows source specific protocols, such as the *Protocol Independent Multicast - Source Specific Multicast* (PIM-SSM) [5], to operate in asymmetric routing environments, without requiring any change to their state machine [3].

This work was supported by the NTT DoCoMo Euro-labs, by the Portuguese Ministry of Science, Technology and High Education, and by European Union FEDER under program POSC.

The tests in this paper were made using PIM-SSM as the underlying multicast routing protocol, and concern both the data and control plane of the OSMAR protocol.

II. RELATED WORK

Many proposals try to solve the problem created by routing asymmetries in IP multicasting, in order to achieve better multicast trees. One of the approaches, the *Yet-Another-Multicast* (YAM) [6], addresses the problem but only for inter-domain multicast branches. Intra-domain branches cannot support routing asymmetries. *Quality of Service-sensitive Multicast Routing Protocol* (QoSMIC) [7] follows YAM in the quest of QoS routing in IP multicast. Although it works both with inter and intra-domain routing asymmetries, its local search method to discover the most suitable path to be used for a new branch, issued from the side of the new member, produces heavy communication overhead, thus being supposed to be only used inside a domain. Additionally, it extends the functionality of PIM-SM and not PIM-SSM. Despite their drawbacks, both YAM and QoSMIC allow receiver initiated, source originating, delivery trees. Other approaches, such as [8], [9] and [10], tried to solve IP multicast problems as a whole by using an application-level multicast system. The goal was to substitute or enhance the network layer multicast support. However, this solutions imply higher complexity and more powerful network nodes, what can prevent its wide use.

III. OSMAR

In this section an overview of OSMAR is presented, together with some details of its implementations in NS2. Implementation of the PIM-SSM protocol is also addressed.

A. *Overview of OSMAR*

OSMAR aims to be used as an overlay for source-specific multicast protocols, like PIM-SSM, enabling them to deal with network asymmetries and to provide QoS-aware content distribution, while maintaining their specifications and state machine. To accomplish this, OSMAR changes the values of the *Multicast Routing Information Base* (MRIB) tables, used by the multicast routing protocol to build the multicast tree, and considers the path from source to receivers. The updated MRIBs will then be used to build optimal trees based on the path that data really uses. In opposition, the normal operation of multicast protocols creates trees following the data path from receivers to the source. Therefore, OSMAR enabled multicast branches are created taking into consideration the connectivity (e.g. unidirectional links) and QoS characteristics configured for each source-receiver data path.

OSMAR works with asymmetric routes, both inside a domain and between domains. Additionally, it allows the adaptation of source-specific multicast protocols to asymmetric routes in a distributed manner, in which most of the functionality is done by OSMAR agents residing in edge routers. Also, OSMAR ensures an easy deployment as it does not require any modification in the state machine of the multicast routing protocol nor any changes to end-hosts. Moreover, OSMAR can be progressively deployed, because it does not require the installation of agents in all network domains.

1) *OSMAR specification*

While OSMAR does not need any changes in the underlying multicast routing protocol it must be able to recognize the IP alert option [11] and to update MRIB tables. Additionally, OSMAR agents in edge routers should also be able to check the PIM *JoinDesired(S,G)* variable status (in order to start OSMAR mechanisms), and the state of each (S,G) channel in each interface.

OSMAR implementation uses three different control messages: *SessAn*, *MribReq* and *MribUp*. The first, *SessAn*, is a multicast message sent by an ingress agent to a well-known multicast group to announce to the local agents that it is the ingress point for the multicast sources specified in the message. It is sent by OSMAR ingress agents at specified time intervals. The second, *MribReq*, is an unicast message sent upstream to request for the update of the MRIBs for any (S,x) channel specified in the message. It can be sent either to build an intra or inter-domain branch, depending if it is sent to an ingress-agent or to the source of the multicast group. In the latter, it can be used together with OSMAR *Fast_Branch* option to indicate that the new branch should be updated from the ingress agent near the source (*Fast_Branch* not set) – the default option -, or from the first OSMAR agent found in the path towards the source that has the specified multicast channel (*Fast_Branch* set). Finally, *MribUp* message is an unicast message sent downstream to update the MRIBs for any channel (S,x) specified in the message. It is sent to the destination *Autonomous System* (AS), where the receiver that initiated the request belongs, and has the IP alert option set.

OSMAR also defines three different functionalities for its agents – ingress, interior, egress-, depending on their locations – ingress router, interior router, egress router. These agents implement all the necessary mechanisms for OSMAR operations.

Only ingress and egress agents (located in edge routers) keep state, and the information is kept for each source-specific channel (S,x). Information kept includes multicast sources, previous egress agents, destination AS, and timers. Interior agents simply change the local MRIBs, not keeping any state.

2) *Detailed functionality of OSMAR*

In the inter-domain, OSMAR helps building QoS multicast tress from the AS where the source is, to the AS where each receiver resides. In the intra-domain, OSMAR helps building QoS multicast trees from the ingress-router of each AS included in the AS tree, to each egress router where receivers have subscribed the session. The way OSMAR helps to build QoS multicast trees relies only in the change of MRIBs, in the path from the source to destination, with the address of a more suitable next-hop router to reach each requested multicast source. This change is done both in inter and intra-domain source-specific multicast branches, independently from the multicast routing protocols. Every MRIB in those branches is updated in a receiver-initiated, source-originated operation.

When a receiver subscribes a source-specific multicast channel, OSMAR egress agent detects the change of the PIM *JoinDesired(S,G)* variable to true. Then, one of two actions is taken, depending on whether the AS where the receiver resides is part of the requested multicast tree. If there is already an ingress agent for the multicast source required in the domain, the egress agent sends a *MribReq* message to that ingress agent, in order to build the intra-domain multicast branch. Otherwise, the egress agent sends a *MribReq* message, with IP alert option, to the multicast source specified by the subscribing receiver, in order to build the inter-domain tree. On receiving the *MribReq* message the receiving agent starts the MRIB updating process, by sending a *MribUp* message towards the destination agent (the one that started the initial request). The path taken by both messages is the path indicated by the available unicast routing protocol.

When building inter-domain multicast trees, the egress agent that initiated the process has the opportunity to select from starting to update the multicast branch from the AS where the source is, or from the first AS with the required multicast channel that is found in the requested inter-domain tree. This choice can be stated by changing the value of the *Fast_Branch* option flag. With this option, the initiating agent can decide for a longer construction time, but a probably more suitable QoS multicast tree, if it chooses to start updating from the AS where the source is. Alternatively it can decide for a shorter construction time, but a probably less suitable QoS multicast tree. In any case, it is the responsibility of the Exterior Gateway Protocol to provide the most suitable route from the agent in the selected AS, until the ingress agent of the AS where the receiver that triggered the update of the inter-domain resides.

Intra-domain multicast branches are updated when an ingress agent for the desirable multicast tree is already known. In this case, the ingress agent in the AS, starts the update of all the MRIBs in the path towards the egress agent that wants to subscribe to the multicast channel. It is the responsibility of the Interior Gateway Protocol to provide the most suitable route to the egress router. It is the responsibility of ingress agents not only to trigger the update of MRIBs in existing intra-domain branches (by sending *MribUp* messages), but also to notify other edge agents in the same domain about the multicast channels that enter the domain at its location (by sending *SessAn* messages).

After receiving the *MribUp* message, the OSMAR agent in the access-router near the receiver allows the continuation of the join process by the multicast protocol.

Fig. 1 shows the request to update the MRIBs for a given (S,x) channel (multicast channel with source S towards any destination x).

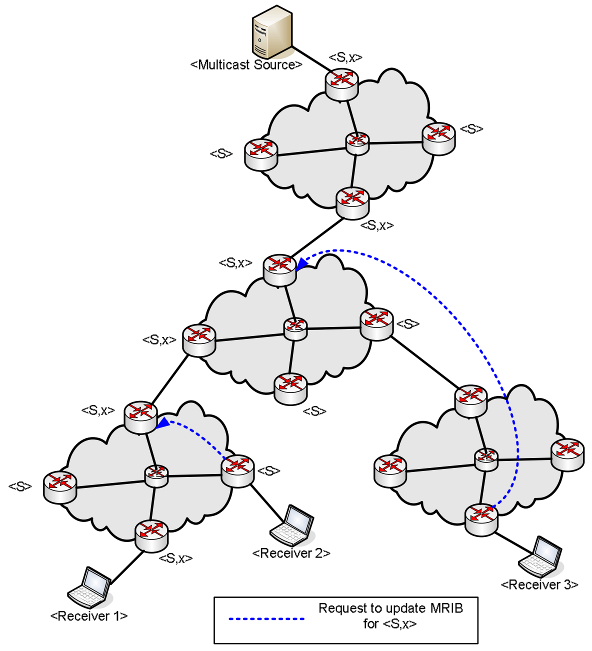

Figure 1. Request to update MRIBs

In this scenario, <Receiver 1> already subscribed to a multicast channel from source S, and so the MRIBs in the path from the source was already updated by OSMAR for (S,x). There are also two egress agents requesting the update of MRIBs, since Receivers 2 and 3 wish to join channels from source S. The egress agent near <Receiver 2> sends a local request, since there is one ingress agent in the domain that has state for (S,x). Finally, the egress agent near <Receiver 3> requests the update of the inter-domain branch since it has no ingress agent in the domain, and chooses to update the multicast branch from the first AS that is found belonging to the inter-domain tree (*Fast_Branch* set).

Having requested the update of MRIBs, OSMAR is now ready to perform the actual update. Fig. 2 depicts how the new branches towards the leaf routers of Receiver 2 and 3 are updated, and how the ingress agents announce to other local agents the channels (S,x) that enter the domain at their location.

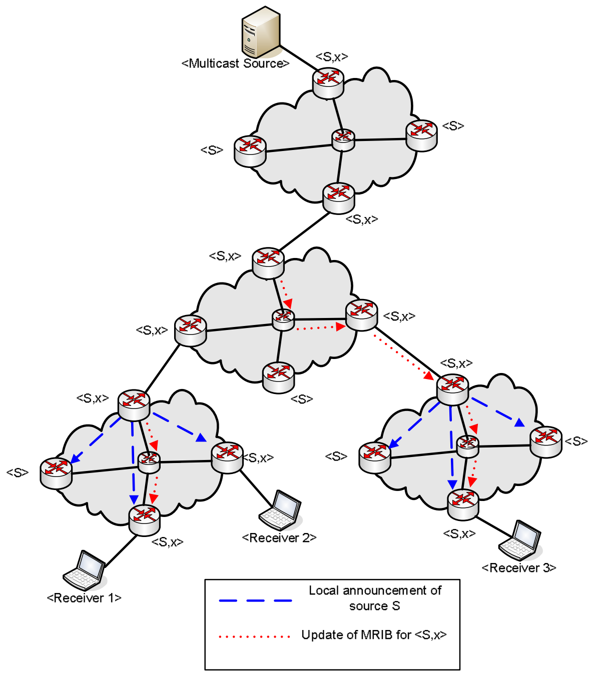

Figure 2. Update of MRIBs and local announcement of source S

If the egress agent near <Receiver 3> had requested the update of the inter-domain branch from the AS where the source is (*Fast_Branch* not set), trying to benefit from a longer but probably more suitable QoS multicast tree, the request would go up to the agent near the <Multicast Source>. Therefore, the following update of MRIBs would be done from the source of the multicast group to the specified receiver.

After OSMAR operation, the updated MRIBs define the best QoS-aware path from source to receiver. The multicast routing protocol can then collect the most suitable next-hop router towards the desired source, from the updated MRIBs. This will allow the construction of a QoS-aware multicast tree.

B. Implementing OSMAR in NS2.

In this simulation, a simplified version of the OSMAR mechanisms described before, was implemented. While OSMAR defines three types of messages to exchange control between agents, in this simulation only *MribReq* and *MribUp* messages were simulated. The reason for not implementing the *SessAn* message, is that its usage, to keep signalling local to access networks, is a clear advantage of OSMAR, and we do not aim to analyse OSMAR in best-case scenarios. The time spent internally to update the MRIB is also not simulated as it is not representative. The size of each OSMAR packet is of 42 bytes: 20 bytes for the *Internet Protocol* (IP) header, 8 bytes for the *User Datagram Protocol* (UDP) header and 14 bytes for OSMAR. These 14 bytes are composed by the OSMAR message type, the address of the previous node in the path, the source address of the multicast channel, the group address to join, and by the *Fast_Branch* field. Although not every field is used in any OSMAR message, the same packet size was used in simulating all messages. Periodically, a timer sends an

MribUp message toward each stored egress agent. This message refreshes the MRIB information in all interior nodes of the path toward each egress agent.

C. Implementing PIM-SSM in NS2

As PIM-SSM is not currently distributed with NS2, a basic implementation [12], was used. This implementation was then revised and extended to support the use of MRIBs. The primary role of the MRIB table in the PIM protocol is to provide the next-hop router along a multicast-capable path to each destination subnet. To populate the MRIB, PIM relies on an underlying topology-gathering protocol [14]. In this simulation, when OSMAR is active, it is the only one that changes the MRIB. When OSMAR is not active, MRIB values are taken from the unicast routing tables.

This new version of PIM-SSM still does not include the exchange of packets necessary to the creation of the multicast tree, nor the exchange of packets used for the subscription to a multicast group by the *Internet Group Management Protocol* (IGMP) [13]. This two features are computed at once and, at the same time, membership is set in the node interfaces for the specific (S,G) channel.

In order to reduce simulation time, the normal 30 s taken by PIM-SSM "Hello" messages, that provide the refresh of multicast trees, was reduced to 0,3 s. PIM-SSM "Hello" message is simulated by a timer that periodically forces the re-computation of multicast trees.

IV. OSMAR EVALUATION

In order to test OSMAR three different scenarios were created. For each scenario, three asymmetry levels were considered - 0%, 30% (average Internet asymmetry level [1]), and 60%. Then, six topologies were randomly created with *Brite* [15] for each scenario (two for each asymmetry level), in a total of 18 topologies. Each of these topologies has 37 nodes (including 1 node that acts like a multicast source and six receivers), and links with a bandwidth of 1 Mb/s. The multicast source sends traffic at a constant bit rate of 500 Kb/s, with a 128 bytes packet size. Asymmetry levels were created by assigning different node delays (and corresponding weights) for traffic travelling in the two different directions of some links. There is no additional traffic in the network.

The first scenario has symmetrical links with a 2 ms delay in both directions. Asymmetric links have 1ms and 3 ms randomly put in each different direction of the link. The second scenario has random delays (between 1 and 4 ms) in each symmetric link, and random delays in each direction of each asymmetric link. This scenario is the most random one. Finally, the third scenario has 2 ms link delay for every link, either symmetric or asymmetric, and random link weights in each direction of asymmetric links. This last scenario corresponds to a network where the routing is only affected by decisions of the network administrators.

In every test there is a multicast source node, and six receiving nodes connected to different points of the network. Each of the six nodes will join the multicast channel during the simulation.

The defined scenarios aimed to evaluate the potential of OSMAR while working together with PIM-SSM, in comparison to a PIM-SSM alone approach. The main objective was to determine if OSMAR allowed PIM-SSM to build more efficient multicast trees, providing a better traffic flow independently of the network conditions. In this analysis, we use as criteria the end-to-end delay, the session setup time, and OSMAR signalling overhead.

In this paper only averages including all scenarios are presented.

A. Impact on data end-to-end delay

This test focus on the data plane, by measuring how much time the data packets take from source to receiver (the multicast tree is already built). Both, PIM-SSM alone and OSMAR + PIM-SSM approaches are used. In the latter case OSMAR is tested with *Fast_Branch* option flag set and unset. The reason for using *Fast_Branch* set and unset is that, in spite of apparently only affecting the time to build the multicast tree, that flag can also lead to the creation of different multicast trees, and therefore producing different delays.

While measuring end-to-end delay, it is clear that OSMAR brings benefits in asymmetric topologies, either with *Fast_Branch* set and unset. These benefits grow along with the asymmetry level as can be seen in Fig. 3, which contains the average of the end-to-end delay measured in all topologies.

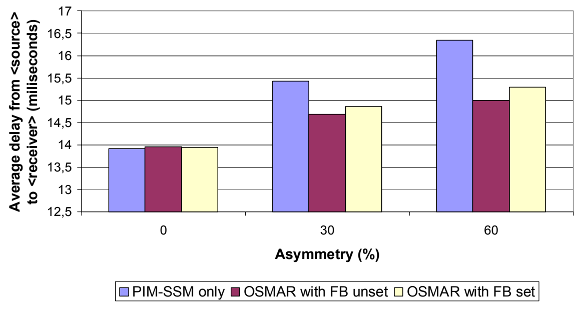

Figure 3. Average delay by asymmetry

By analysing results in more detail it is found that, in most cases, OSMAR with the *Fast_Branch* flag unset leads to lower end-to-end delays. This may be explained by the fact that without this option the tree is built from the multicast source, and not from an intermediary branch. This difference leads to the construction of multicast trees with lower end-to-end delay since all the possible links from source to destination are considered.

For a better analysis of how much faster is OSMAR when compared with PIM-SSM, we calculated the speedup of OSMAR (with *Fast_Branch* flag unset) vs PIM-SSM. The speedup, is the ratio between the average delay achieved by PIM-SSM alone and the average delay achieved when using OSMAR. The speedup averages 1.0508 in 30% asymmetry level topologies, and 1.0907 in 60% asymmetry level topologies. In 0% asymmetry level topologies, a small decrease in the performance of OSMAR, relatively to a PIM-SSM approach can be detected. This is due to the small overhead of the OSMAR control messages - the transmission of *MribUp* messages in response to *MribReq* messages, and the *MribUp* messages that OSMAR refresh implies, results in a slight delay of some CBR data packets. However, the difference is minimal, averaging more 0.035 ms than when using PIM-SSM alone.

B. *Impact on session setup*

This test measured the time taken by OSMAR messages to make the necessary changes to the MRIBs. Time is measured from the initial *MribReq* message sent when a node wants to join the multicast channel (S,G), until the time when the corresponding *MribUp* message is received by the initial joining node. Results using OSMAR with and without the *Fast_Branch* option are displayed in Fig. 4.

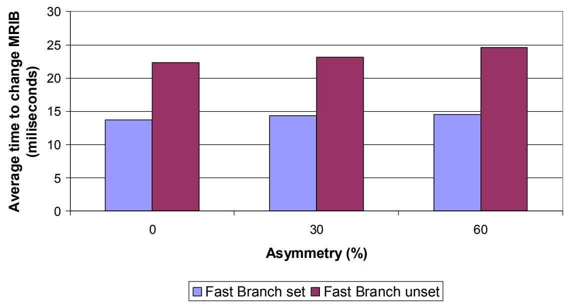

Figure 4. Average MRIB changing time by asymmetry

As shown in Fig. 4, which includes all tests done in every topology, average times to change MRIBs are slightly bigger with higher asymmetry levels, both with and without *Fast_Branch* option flag set. The measured average setup time, due to the time that OSMAR needs to change the MRIB in all routers, ranges from 13.686 ms (0% symmetry scenario) to 14.534 ms (60% symmetry scenario) with *Fast_Branch* option set, and from 22.273 ms (0% symmetry scenario) to 24.574 ms (60% symmetry scenario) with *Fast_Branch* option unset.

As could be forecasted, the *Fast_Branch* field set option shows clear benefits in what respects the setup time, since with the *Fast_Branch* field OSMAR starts configuring the MRIBs closer to the receivers.

Based on this test, we can also conclude that the asymmetry levels of the different topologies scenarios do not influence the results significantly.

C. *Impact on session setup (optimized)*

One of the drawbacks of using OSMAR is the initial time spent by OSMAR to update the MRIBs, which delays the start of the reception of data by the receiver. An alternative approach is to permit the occurrence of a normal multicast join and, at the same time, the start of OSMAR operations. I.e., OSMAR does not block the multicast join request, and so, the join is made before OSMAR finishes changing all MRIBs. Hence, the multicast trees built by PIM-SSM use two types of MRIB values: first, during tree setup, the MRIB values based on the unicast routing table, and after that, during tree refreshment, using the MRIB values updated by OSMAR. In this test (without *Fast_Branch*) the number of out-of-order packets that arrive to the receiver, and of data packets lost in the transition from the original PIM-SSM built tree to the new OSMAR changed tree, are measured.

Simulation results show that the average number of lost packets to each of the receiver access routers, is minimal.(from an average of 0.97 packets in 0% asymmetries to 1.75 in 60% asymmetries). Out-of-order packets do not surpass one packet (it is the first packet received after the lost ones, therefore being out-of-order).

In the interpretation of these results, it should be noted that due to the fact that the PIM-SSM implementation does not have packet exchange, therefore being immediate either in the first time it is used, as in consecutive refreshes, there are packets that are caught in the middle of their path when the multicast tree is changed. This results in situations where the packets suddenly do not have a path to follow, and therefore are lost. With a packet exchange implementation, the overall number of lost packets should be less that the ones resulting from this simulation.

D. *Signalling overhead*

The main objective of this test was to measure the overhead produced by OSMAR in the network, and its scalability.

The packets that are accounted for, are the OSMAR packets sent by each node (that simulates a router) in the network during the simulation, either being the initial sending node or any of the routers that forward the packet through the network. In Fig. 5 the average number of OSMAR packets in the network is displayed by asymmetry (includes all scenarios). As would be expected, the number of OSMAR packets accounted in all links is higher when *Fast_Branch* option flag is not set, as the packets have to travel the complete path between each receiver access router to the multicast source. The average number of OSMAR packets in the network varies slightly with asymmetry, reflecting the different number of links in the many multicast trees tested - as the tree has one more link there is one more OSMAR control message (travelling downstream) set to the network through that specific link. It should be noticed that the simulation set the OSMAR refresh timer to 0.3 s, while it is expected to be close to the value of PIM-SSM refresh messages that is 30s.

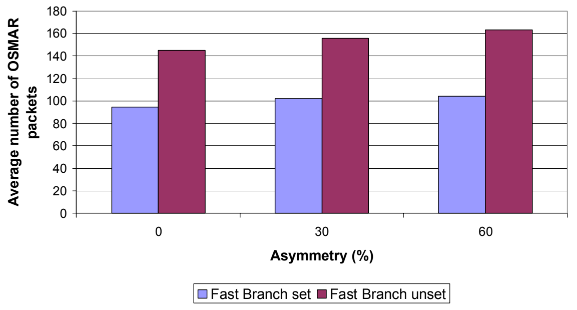

Figure 5. Average number of OSMAR packets in the network by asymmetry

The distribution of the average number of OSMAR and CBR packets along all the simulation time is displayed in

Fig. 6 (*Fast_Branch* not set). This average is the average of packets in all topologies tested. In this simulation, receiver access routers signal their intention to join the multicast channel at instants 0.1 s, 0.3 s, 0.5 s, 0.7 s, 0.9 s, and 1.1 s.

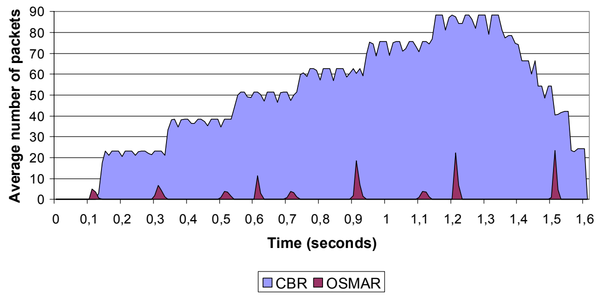

Figure 6. Average number of OSMAR packets in the network during the simulation, with *Fast_Branch* not set

When the first receiver joins the multicast channel, OSMAR is triggered leading to a small bump in the number of OSMAR packets in the network. The reception of *MribUp* messages signals the access-node that the multicast join can be completed, as the MRIBs are already changed (the optimized version analysed in section IV-C is not used). This leads to the growing of CBR packets in the network, as multicast CBR traffic is delivered to the different receivers. The successive joins lead to successive increases in the number of CBR packets in the network, that only decreases in the end of the simulation as nodes leave the multicast channel. Each 0.3 seconds there is an increase of OSMAR packets in the network, which corresponds to OSMAR refresh messages (the ones that permit the maintenance of the MRIB tables).

As for scalability, we can observe that the number of OSMAR packets grows linearly with the number of receivers, and can even be reduced by using the *Fast_Branch* field set, and by separating the topology in different domains, in which the OSMAR announcement mechanism is used. It should also be noticed that the bandwidth occupied by OSMAR does not relate to the CBR data packets bandwidth, as it only depends on the number of receivers that join the multicast channel.

V. CONCLUSION

The need of QoS support in multicast delivery of multimedia contents is growing. However, most of today's multicast routing protocols do not provide for any QoS restraints in the building of multicast trees, and do not take into account routing asymmetries. In this paper we present OSMAR, an overlay that allows enhancing PIM-SSM trees with QoS awareness in networks with routing asymmetries, without changing standards.

In the simulations, OSMAR was tested in data and control plane, using PIM-SSM multicast routing protocol. The aim was to find out if OSMAR could provide a better overall performance.

As result of all the tests made it can be concluded that in the presence of asymmetries in the network, OSMAR brings clear advantages, by averaging a lower end-to-end packet delay with a negligible overhead.

The drawback initially expected from OSMAR, the overhead that it would create in the session setup, was avoided by the use of an optimized version for the initial MRIBs update. This alternative approach, which enables simultaneous PIM-SSM joins and OSMAR MRIBs update, proved to be more efficient than the initially proposed, achieving a performance equal to PIM-SSM despite a small cost in the amount of lost packets. Also, OSMAR packet overhead in the network was found to grow linearly with the number of receivers, while consuming a negligible portion of the bandwidth.

After analyzing all the results, it can be said that OSMAR allows PIM-SSM to build better multicast trees in scenarios where routing asymmetries exist. By averaging a lower end-to-end delay, it enables traffic to travel through the more advisable links, as the network managers characterized them (as it follows lower cost links), providing a QoS-awareness protocol that PIM-SSM by itself can not guarantee.

REFERENCES

[1] V. Paxson, "End-to-End Routing Behavior in the Internet", IEEE/ACM Transactions on Networking, Vol.5, No.5, pp. 601-615, October 1997.

[2] Y. He, M.Faloutsos, S.Krishnamurthy, B.Huffaker, "On routing asymmetry in the Internet", IEEE GLOBECOM '05, St.Louis.

[3] P. Mendes, "OSMAR: Overlay for Source-specific Multicast in Asymmetric Routing environments", Technical Report, NTT DoCoMo Euro-Labs, 2004.

[4] "The Network Simulator - ns-2": http://nsnam.isi.edu/nsnam.

[5] H. Holbrook and B. Cain, "RFC 4607: Source-Specific Multicast for IP": Internet Engineering Task Force, August 2006.

[6] Ken Carlberg, Jon Crowcroft, "Building Shared Trees Using a One-to-Many Joining Mechanism", ACM SIGCOMM Computer Communication Review, Vol.27, January 1997.

[7] Michalis Faloutsos, Anindo Benerjea, Rajesh Pankaj, "QoSMIC: Quality of Service Sensitive Multicast Internet Protocol", In Proc. of ACM SIGCOMM, Vancouver, Canada, September 1998.

[8] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang, "A Case for End System Multicast," IEEE Journal on Selected Areas in Communications, vol. 20, pp. 1456-1471, October 2002

[9] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, J. O'Toole, "Overcast: Reliable Multicasting with an Overlay Network", 4th OSDI, Dec 2000.

[10] S. Shi and J. Turner, "Routing in Overlay Multicast Networks", INFOCOM 2002.

[11] D. Katz, "RFC 2113: IP Router Alert Option": Internet Engineering Task Force, February 1997.

[12] T. Camilo, J. Sá Silva, F. Boavida, "Modulo SSM para Ambientes Simulados em NS-2", in Proc. of the Actas da CRC'2004 – 7ª Conferência sobre Redes de Computadores, October 2004..

[13] B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 2236, Internet Engineering Task Force, October 2002.

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

[15] "brite: Boston University Representative Internet Topology Generator".