Enabling the Progressive Multicast Service Deployment

Luís Henrique M. K. Costa[1,2], Serge Fdida[1]
[1] Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie
4, place Jussieu - 75252
Paris Cedex 05 - France
Luis.Costa@lip6.fr, Serge.Fdida@lip6.fr

Otto Carlos M. B. Duarte[2]
[2] Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro
P.O. Box 68504 - 21945-970
Rio de Janeiro - RJ - Brasil *
otto@gta.ufrj.br

Abstract

The IP Multicast architecture was not widely deployed because multicast address allocation is difficult and there is no scalable solution to inter-domain multicast routing. Hence, there is an interest in developing protocols that allow the progressive deployment of the multicast service by supporting unicast clouds. This paper proposes HBH (Hop-By-Hop multicast routing protocol). HBH adopts the source-specific channel abstraction to simplify address allocation and implements multicast distribution using recursive unicast trees. In this model, data packets have unicast destination addresses. Therefore, HBH supports pure unicast routers transparently. The branching-nodes recursively create packet copies to implement the distribution. HBH constructs a Shortest-Path Tree even in the presence of asymmetric unicast routing. Consequently, HBH provides best routes in asymmetric networks, and is suitable for an eventual implementation of QoS-based routing. Additionally, HBH reduces tree bandwidth consumption in asymmetric networks when compared to other approaches. The results obtained from simulation support our statements.

1. Introduction

The current IP Multicast architecture did not become ubiquitous, and it is not likely to see its wide implementation in the near future. This is mainly due to the very definition of IP Multicast service model provides open access to the information [3]. Any host can send to a multicast group and any host can join this group and receive data from it. The IP Unicast service model is also completely open, but with multicast the potential burden caused by unauthorized senders is amplified by the size of the group.

The second part of IP Multicast includes group addressing and routing protocols. A group is identified by a class-D IP address which is not related to any topological information, as opposed to the hierarchical unicast addresses. Therefore, multicast address allocation is complicated and multicast forwarding state is difficult to aggregate. No scalable multicast routing protocol was yet defined. As a consequence, the research community started to think about alternative models to multicast data distribution.

Some proposals tried to simplify the multicast service [6]. The analysis of these works lead us to the proposition of the Hop-By-Hop multicast routing protocol (HBH). HBH borrows its basic ideas from EXPRESS [11] and REUNITE [12]. EXPRESS reduces the distribution model from M to N to 1 to N, simplifying the service. REUNITE implements multicast distribution through a recursive unicast tree, being able to support pure unicast routers. Both protocols are discussed in more detail in Section 2.

HBH uses the unicast infrastructure to do packet forwarding with smaller routing tables, just as REUNITE does, but uses EXPRESS' channel model to identify a group. Thus HBH preserves compatibility with IP Multicast. HBH constructs Shortest-Path Trees (SPT) instead of Reverse SPTs as most routing protocols do [13, 4, 7, 5]. Consequently, HBH provides best routes in asymmetric networks. Additionally, HBH's tree management provides enhanced tree stability in the presence of group dynamics and reduces bandwidth consumption in asymmetric networks.

This paper is organized as follows: Section 2 presents the motivations of HBH, Section 3 HBH functioning, and Section 4 the comparison of HBH and other protocols through simulation. Section 5 concludes the paper.

2. Motivations

EXPRESS provides a simple solution to the multicast addressing problem, introducing the channel abstraction that restricts the distribution to *1 to N*. A channel is identified by the pair $<S, G>$ where S is the unicast address of the source and G is a class-D multicast address. The concatenation of an unicast address with a class-D address solves

*This work was sponsored by FUJB, CNPq, CAPES/COFECUB, and IST Project GCAP N^0 1999-10504.

the address allocation problem since the unicast address is unique. The model also simplifies group management issues such as sender access control, although its implementation (PIM-SSM [10]) adds no group management support.

REUNITE (REcursive UNIcast TrEes) [12] implements multicast distribution based on the unicast routing infrastructure. The basic observation is that in typical Internet multicast trees, most routers simply forward packets from one incoming interface to one outgoing interface, in other words, the minority of routers are branching nodes. Nevertheless, all multicast protocols keep per group information at all routers in the tree. The idea is then to separate the routing information into two tables: a Multicast Control Table (MCT) stored in the control plane and a Multicast Forwarding Table (MFT) installed in the data plane. Non-branching routers simply keep group information in their MCT, as branching nodes keep MFT entries used to recursively create packet copies that reach all group members.

REUNITE identifies the group by a $<S, P>$ tuple, where S is the unicast address of the source and P is a port number allocated by the source. Class-D IP addresses are not used. As receivers join the group REUNITE populates its tables to construct the distribution tree. REUNITE uses two message types: *join* messages travel upstream from the receivers to the source, as *tree* messages are periodically multicast by the source to refresh the soft-state of the tree.

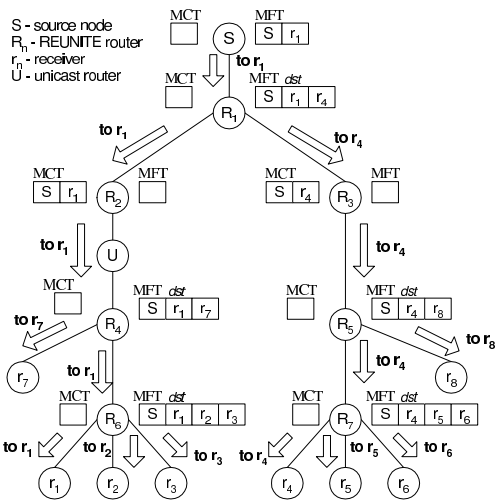

Figure 1. REUNITE tree.

Multicast distribution is implemented through recursive unicast. The source sends data in unicast to the first receiver that joined the group. At a branching node, R_B, entering data packets are addressed to the first receiver, r_i, that joined the group in the sub-tree below R_B. r_i is stored in a special MFT entry, MFT<S>.dst. R_B creates one packet copy for each receiver in its MFT (the destination address is set to the receiver's unicast address). The original packet is also forwarded to r_i. Figure 2 gives an example. S produces data packets addressed to r_1 (these packets reach r_1 unchanged). R_1 creates one copy and sends it to r_4. Since R_3 is a non-branching node, it simply forwards the packets without consulting its MFT. R_5 creates one copy to R_8 and finally R_7 sends one copy to r_5 and r_6.

REUNITE allows the progressive multicast deployment, because the use of unicast addresses turns the protocol able to support unicast routers in the distribution tree. These routers are not able to be branching nodes but can forward data packets since they have unicast destination addresses.

Nevertheless, REUNITE has problems if unicast routing is asymmetric. Its tree is a SPT if unicast routes are symmetric but this can be false with asymmetric networks. Receivers may be connected to the distribution tree through a path different from the shortest path. A second undesirable behavior of REUNITE is that the route for a certain receiver may change after the departure of another receiver.

2.1. REUNITE and asymmetric routing

Asymmetric unicast routing affects multicast routing since the majority of multicast routing protocols construct *Reverse* Shortest-Path Trees [13, 4, 7]. Data packets from the source to a receiver follow the unicast route used to go from the receiver to the source. These paths may have different characteristics, e.g. different delays. The ability to construct Shortest-Path Trees is therefore advantageous to the multicast routing protocol.

One of the characteristics that differentiates REUNITE from previous routing protocols is its ability to construct SPTs. This is possible because the *tree* messages sent from the source to the destination are responsible for tree construction and not the *join* messages that follow the reverse path. Nevertheless, REUNITE may construct non shortest-path branches if unicast routing is asymmetric.

Figure 2 illustrates REUNITE tree construction mechanism and gives an example where it fails to construct a SPT. Suppose the unicast routes: r_1>R_2>R_1>S ; S>R_1>R_3>r_1 ; r_2>R_3>R_1>S ; S>R_4>r_2. Suppose the following events: r_1 joins <S, P>, r_2 joins <S, P>, and r_1 leaves the group.

r_1 subscribes to the channel by sending a $join(S, r_1)$[1] message to S. This message reaches S since there is no previous tree state. We say that r_1 *joined* <S, P> at S. S then starts sending $tree(S, r_1)$ messages to r_1 (in unicast). These *tree* messages install soft-state for <S, P> in the routers traversed downstream. R_1 and R_3 create a <S, r_1> entry in their MCT. Now r_2 joins the group. The $join(S, r_2)$ travels to S reaching the tree at R_3. R_3 drops the $join(S, r_2)$, creates a MFT<S> with r_1 as *dst*, adds r_2 to MFT<S>, and removes <S, r_1> from its MCT. R_3 becomes a branching node and will consequently forward

[1] In the rest of the paper, we interchangeably use <S> and <S, P> to refer to the multicast channel.

$tree(S, r_2)$ messages downstream (upon the reception of $tree(S, r_1)$). We say that r_2 joined the channel at R_3. Subsequent *join* messages sent by r_1 and r_2 refresh the MFT entries at S and R_3 respectively.

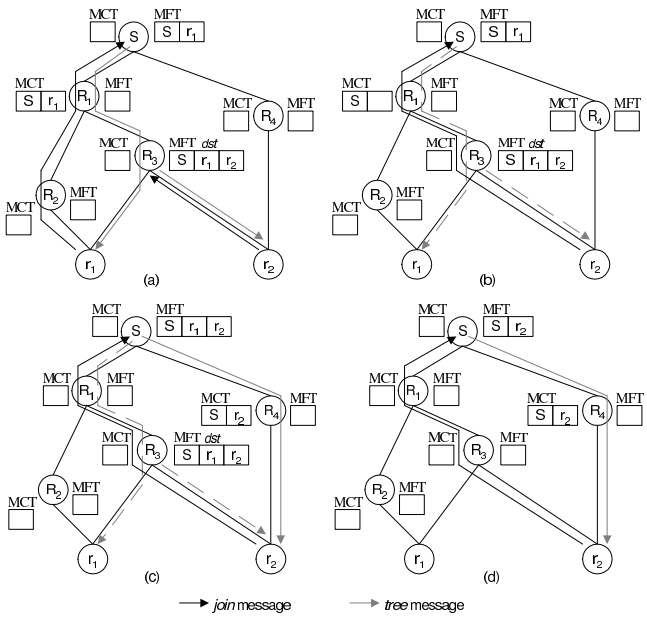

Figure 2. REUNITE's tree construction.

In this configuration, r_1 receives data from S through the shortest-path, but not r_2. Because the unicast routes between S and r_2 are asymmetric and because R_3 intercepts $join(S, r_2)$, data follows the path S>R_1>R_3>r_2, the same as *tree* messages from S down to r_2 (Figure 2(a)).

MCT and MFT states are soft. Receivers periodically send $join(S, r_i)$ messages and the source periodically multicasts a $tree(S, r_i)$ message. The receiver simply stops sending *join* messages to leave the channel. When the tree structure is stable, a $tree(S, r_i)$ message refreshes the r_i MCT entries and the MFT.*dst* = r_i entries down the tree. The $join(S, r_j)$ messages refresh the r_j entry in the MFT of the node where r_j joined <S>.

Now r_1 leaves the group: it stops sending $join(S, r_1)$ messages. As the r_1 entry in S's MFT is not refreshed, after the expiration of timer *t1* the r_1 entry becomes stale. A second timer, *t2*, is created and will eventually destroy the r_1 entry. As r_1 is stale, S now sends marked $tree(S, r_1)$ messages(Figure2(b)). The marked $tree(S, r_1)$ means that data flow addressed to r_1 will stop soon, so the tree portion based on r_1 has to be reconfigured. At the branching nodes, MFT tables that have MFT<S>.$dst = r_1$ become stale as the marked $tree$ travels down the tree. At non-branching nodes, the reception of a stale $tree(S, r_1)$ causes the destruction of any r_1 MCT entries. Consequently, $join(S, r_2)$ messages are no more intercepted by R_3 (as its MFT<S> is stale) and reach S. r_2 now joins <S, P> at S (Figure2(c)). Eventually *t2* timeouts result in the deletion of r_1 from S's and R_3's MFTs. As R_3 stops receiving *tree* messages, its MFT is destroyed (Figure 2(d)). In this new configuration, r_2 receives data through the shortest-path from S.

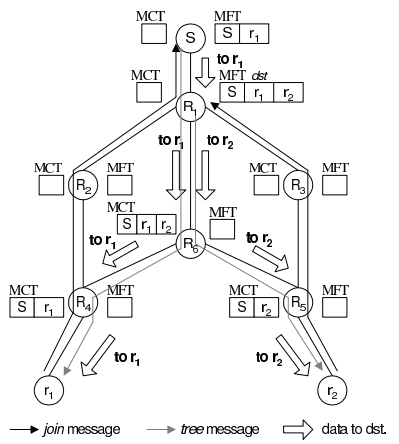

Figure 3. Packet duplication in REUNITE.

Asymmetric routing may also lead REUNITE to unneeded packet duplications.[2] Figure 3 gives an example. The first receiver, r_1, sends a $join(S, r_1)$ that follows the path r_1>R_4>R_2>R_1>S. The $tree(S, r_1)$ messages follow the route S>R_1>R_6>R_4>r_1. Suppose now that r_2 joins and that $join(S, r_2)$ follows r_2>R_5>R_3>R_1. The $tree(S, r_1)$ (produced by S) and the $tree(S, r_2)$ (created at R_1) both traverse the link R_1-R_6. As R_6 does not receive $join$ messages from these receivers, it is not identified as a branching node. S creates data packets to r_1 and R_1 creates packets to r_2. So there is two packet copies on the link R_1-R_6. Consequently, the cost of a REUNITE tree may be greater than that of a source tree constructed by a classic protocol as PIM-SM [7]. The next Section describes HBH and how its tree construction mechanism is able to cope with the problems due to asymmetric unicast routing.

3. Hop-By-Hop Multicast Protocol

HBH uses two tables, one MCT and one MFT that have nearly the same function as in REUNITE. The difference is that one entry table in HBH stores the address of a *next branching node* instead of the address of a *receiver* (excepted for the branching router nearest the receiver). The MFT has no *dst* entry. Data received by a branching router, H_B, has unicast destination address set to H_B (as opposed to MFT.<*dst*> in REUNITE). This choice turns the tree structure more stable than in REUNITE. A multicast channel in HBH is identified by <S, G>, as in EXPRESS. This definition solves the multicast address allocation problem

[2] In fact, this possibility also exists when the network has pure unicast routers or when the REUNITE router is overloaded [12].

while being compatible with with IP Multicast, because HBH can potentially support IP Multicast clouds as leaves of the distribution tree.

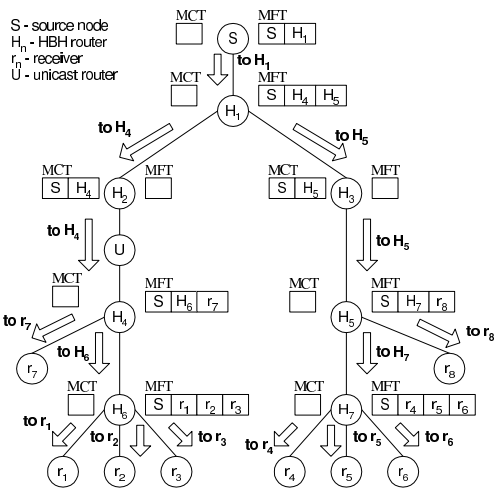

Figure 4. HBH tree.

Figure 3 shows a HBH tree. S sends data addressed to H_1. H_1 creates two packet copies and sends them to H_4 and H_5 (the next branching nodes). H_3 simply forwards the packets in unicast. H_5 receives the data and sends a modified packet copy to H_7 and r_8. Finally, H_7 creates one packet copy to r_4, r_5, and r_6. Data distribution is symmetric on the other side of the tree.

HBH's tree structure has the advantage of an enhanced stability of the table entries when compared to REUNITE. The tradeoff is that in HBH each data packet received by a branching node produces $n + 1$ modified packet copies while in REUNITE it produces n modified packets. The tree management scheme of HBH minimizes the impact of member departures in the tree structure. This is possible because the MFT receiver entry is located at the branching node nearest the receiver.

3.1. Tree management in HBH

HBH has three message types: $join$, $tree$, and $fusion$. $Join$ messages are periodically unicast by the receivers in the direction of the source and refresh the forwarding state (MFT entry) at the router where the receiver joined. A branching router "joins" the group itself at the next upstream branching router. Thus the $join$ messages may be intercepted by the branching nodes which sign themselves $join$ messages. The source periodically multicasts a $tree$ message that refreshes the rest of the tree structure. $Fusion$ messages are sent by potential branching routers and construct the distribution tree together with the $tree$ messages.

Each HBH router in S's distribution tree has either a MCT<S> or a MFT<S>. A non-branching node in S's distribution tree has a MCT<S>. MCT<S> has one single entry to which two timers are associated, *t1* and *t2*. At the expiration of *t1* the MCT becomes stale and at the expiration of *t2* the MCT is destroyed.

A branching node in S's distribution tree has a MFT<S>. Two timers, *t1* and *t2*, are associated to each entry in MFT<S>. When *t1* times out the MFT entry becomes stale and it is destroyed when *t2* expires. In HBH, a *stale* entry is used for data forwarding but produces no downstream $tree$ message. A MFT entry in HBH can also be *marked*. A *marked* entry is used to forward $tree$ messages but not for data forwarding.

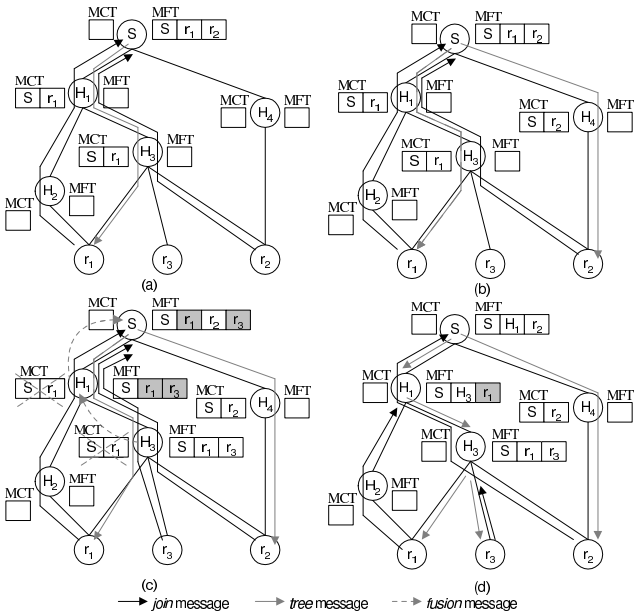

Figure 5. HBH's tree construction.

We will illustrate HBH functionning through an example (a detailed description of the message processing rules can be found in [2]). Figure 5 repeats the scenario of Figure 2. r_1 joins the multicast channel at S that starts sending $tree(S, r_1)$ messages. These messages create a MCT<S> containing r_1 at R_1 and R_3 (Figure 2(a)). When r_2 joins the group by sending the first $join(S, r_2)$, this message is not intercepted and reaches S (the first $join$ message is never intercepted). The $tree(S, r_2)$ produced by the source create MCT<S> state at R_4 (Figure 5(b)). Both receivers are connected to the source through the shortest-path.

Suppose now that r_3 (unicast routes: S>R_1>R_3>r_3 and r_3>R_3>R_1>S) joins the channel. It sends a $join(S, r_3)$ to S that starts sending $tree(S, r_3)$ messages. As R_1 receives two different $tree$ messages, it sends a $fusion(S, r_1, r_3)$ to the source. The reception of the $fusion$ causes S to mark the r_1 and r_3 entries in its MFT and to add R_1 to it. In the same way as R_1, R_3 receives $tree(S, r_1)$ and $tree(S, r_3)$ messages and thus send a $fusion(S, r_1, r_3)$ to the source (Figure 5(c)). R_3's MFT now contains r_1 and r_3. Subse-

quent $join(S, r_1)$ messages are intercepted by R_1 and refresh the r_1 marked entry in R_1's MFT. The $join(S, r_3)$ messages refresh the r_3 MFT entry at R_3. S sends data addressed to R_1, that sends it addressed to R_3. R_3 sends copies to r_1 and r_3. Subsequently, as S receives no more $join(S, r_1)$ neither $join(S, r_3)$ messages, its corresponding MFT entries are destroyed. The final structure is shown in Figure 5(d). In this way, HBH is able to use the good branching point to the distribution tree. The problem of Figure 3 is simply resolved through the transmission of a $fusion(S, r_1, r_2)$ from R_6 upstream to the source, similarly to the example just presented.

4. Performance Analysis

We used NS (Network Simulator)[8] to simulate HBH. Our objectives were to experiment HBH's tree mechanisms as well as to compare HBH and REUNITE through the analysis of the constructed trees. We analyzed the average delay experienced by all the receivers of the group and the number of packet copies that are needed to reach all receivers.

The topology used in our simulations is typical of a large ISP's network [1]. Without loss of generality, we suppose that only one receiver is connected to each node in the topology. The presence of one or many receivers attached to a border router through IGMP [9] does not influence the cost of the tree, so we do not consider the aggregation provided by the multicast service at the local network level.

We associate two costs, $c(n_1, n_2)$ and $c(n_2, n_1)$, to link n_1-n_2. Each cost is an integer randomly chosen in the interval $[1, 10]$. Simulations consider one multicast channel with fixed source. A variable number of randomly chosen receivers join the channel. For each number of receivers we realized 500 simulation runs per protocol.

4.1. Results

We compared HBH to REUNITE and two classical multicast approaches that are available in NS. NS has a multicast routing protocol that is able to construct shared trees and source trees with the same structure as the trees constructed by the PIM-SM protocol. The difference is that NS's implementation is centralized and the change from the shared tree to the source tree is realized through an explicit command, and not automatically as in the original PIM-SM [7]. Therefore, PIM-SM in our simulations refers to a protocol that constructs exclusively shared trees, whereas PIM-SS is a protocol that only constructs source trees. The tree structure of PIM-SS is the same as that of PIM-SSM [10], i.e., a reverse SPT. In addition to HBH, we implemented REUNITE according to [12]. All routers implement the multicast service in our experiments.

Tree cost - We define the cost of a tree as the number of copies of the same packet that are transmitted in the network links and not as the number of links in the tree because in the recursive unicast technique more than one copy of the same packet may be sent over one link. This may be due to the network's routing asymmetries (Section 2.1) but also to unicast routers that can not be branching nodes. Nevertheless, as in our experiments all routers are multicast capable, extra packet copies are always due to routing asymmetries.

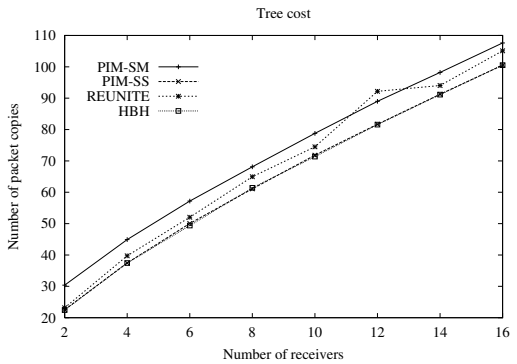

Figure 6. Tree cost comparison.

Figure 6 shows the average cost of the trees as the number of receivers varies. PIM-SM constructs the trees with the highest cost in most cases. This is expected since PIM-SM constructs shared trees. As we simulated the distribution from one source to many receivers, the utilization of a shared tree is disadvantageous since the tree is centered on a *rendez-vous* point (RP). This tree probably has a higher cost than the equivalent source tree. HBH and PIM-SS construct the cheapest trees. PIM-SS constructs source trees based on the RPF algorithm. It guarantees that at most one copy of the same packet is transmitted at each link and on that each receiver is connected to the source trough the *reverse* shortest-path. HBH has a similar performance because it connects each receiver to the source through the shortest-path. Using the shortest path from source to receiver or from receiver to source is equivalent.

Figure 6 shows that REUNITE suffers from the pathological cases produced by asymmetric unicast routing. The phenomenon is less frequent with a small number of receivers, since the probability that two receivers share the same link in the multicast tree is smaller. The problem is also less severe when the number of receivers is huge since most of the network links are used in the tree. HBH has a more efficient tree construction mechanism. In terms of tree cost, the advantage of HBH over REUNITE is as large as 5% in average. This difference means that HBH provides a better bandwidth utilization than REUNITE.

Delay - Figure 7 presents the average delay experienced by the receivers. Two results are unexpected. First, PIM-SM performs better that PIM-SS (i.e. , the shared trees have better delay than the source trees). This is because PIM-SS tree is a *reverse* SPT and not a SPT. So delay is not minimized. Delay is not minimized either in the PIM-SM

shared tree, but the paths from the source to the receivers all have one part in common, between the source and the RP. As data is encapsulated in unicast between the source and the RP, delay is minimized. Thus, paths in the PIM-SM tree have two parts: from the source to the RP where delay is minimized and from the RP to the receiver where it is not minimized (it is a *reverse* shortest path). The second important remark is that the effect of the network asymmetries in the quality of REUNITE trees may be strong, as it performed worse than PIM-SM for large receiver sets.

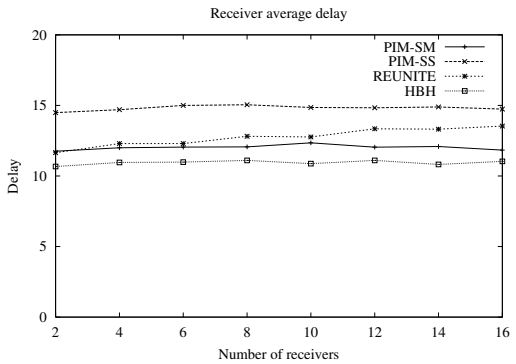

Figure 7. Receiver delay comparison.

HBH is effectively able to generate better quality routes than REUNITE in the presence of asymmetric unicast routing. The performance of HBH is better than that of REUNITE for all group sizes. The advantage becomes larger as the number of receivers grows, being of 14% in average.

Considering the network size (18 nodes with 3.3 connectivity) HBH's advantage to REUNITE (5% in cost and 14% in delay) is a promising result. This difference should grow with the network size as the probability of routing asymmetries becomes larger.

5. Conclusions

We presented HBH, a multicast routing protocol that implements data distribution through recursive unicast trees, idea originally proposed in REUNITE [12]. HBH allows the progressive deployment of the multicast service as unicast routers are transparently supported. The objectives of HBH are to go through unicast clouds; to have a stable tree structure in the presence of to member departures; to provide low cost trees; and to guarantee that data is received through the shortest path from the source.

HBH has an original tree management algorithm that is based on three messages. *Join* messages are periodically sent to the source by the receivers. The source periodically produces *tree* messages that are multicast to the receivers. As the *tree* messages travels in the tree the intermediate nodes may generate *fusion* messages that are responsible of refining the tree structure.

HBH is able to construct a Shortest-Path Tree even if unicast routing is asymmetric and provides better network utilization as it constructs trees minimize packet duplication. Additionally, HBH's delay performance makes it adapted to applications that do not support large delays such as interactive ones.

The results obtained through simulation show that HBH is a promising approach as it outperforms REUNITE by 5% in terms of tree cost and 14% in terms of the delay experienced by the receivers. Our future work includes experiments with larger networks and unicast clouds. We will also formally define the interface between HBH and classical IP Multicast, and study the possibility of including QoS parameters inside HBH's tree construction algorithm.

References

[1] G. Apostolopoulos, R. Guerin, S. Kamat, and S. K. Tripathi. Quality of service based routing: A performance perspective. In *ACM SIGCOMM'98*, pages 17–28, Sept. 1998.

[2] L. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte. *Hop By Hop Multicast Routing Protocol*. RP-LIP6 Technical Report available at http://www-rp.lip6.fr/publications, Feb. 2001.

[3] S. Deering. *Host Extensions for IP Multicasting*. RFC 1112, Aug. 1989.

[4] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C.-G. Liu, and L. Mei. The PIM architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2):153–162, Apr. 1996.

[5] C. Diot, W. Dabbous, and J. Crowcroft. Multipoint communication: A survey of protocols, functions and mechanisms. *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, Apr. 1997.

[6] C. Diot, B. N. Levine, B. Liles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. *IEEE Network*, pages 78–88, Jan. 2000.

[7] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. *Protocol Independent Multicast-Sparse Mode (PIM-SM): Protocol Specification*. RFC 2362, June 1998.

[8] K. Fall and K. Varadhan. *The ns Manual*. UC Berkeley, LBL, USC/ISI, and Xerox PARC, Jan. 2001. Available at http://www.isi.edu/nsnam/ns/ns-documentation.html.

[9] W. Fenner. *Internet Group Management Protocol, Version 2*. RFC 2236, Nov. 1997.

[10] H. Holbrook and B. Cain. *Source-Specific Multicast for IP*, Mar. 2001. Internet-draft: draft-holbrook-ssm-02.txt.

[11] H. W. Holbrook and D. R. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. In *ACM SIGCOMM'99*, Sept. 1999.

[12] I. Stoica, T. S. E. Ng, and H. Zhang. REUNITE: A recursive unicast approach to multicast. In *IEEE INFOCOM'2000*, Mar. 2000.

[13] D. Waitzman, C. Partridge, and S. Deering. *Distance Vector Multicast Routing Protocol*. RFC 1075, Nov. 1988.