Hop By Hop Multicast Routing Protocol

Luís Henrique M. K. Costa[1,2] Serge Fdida[1] Otto Carlos M. B. Duarte[2]
Luis.Costa@lip6.fr Serge.Fdida@lip6.fr otto@gta.ufrj.br

[1] Laboratoire d'Informatique de Paris 6 / Université Pierre et Marie Curie
4, place Jussieu - 75252 - Paris Cedex 05 - France

[2] Grupo de Teleinformática e Automação / Universidade Federal do Rio de Janeiro
P.O. Box 68504 - 21945-970 - Rio de Janeiro - RJ - Brasil *

Abstract

IP multicast has been addressed since more than a decade but very little has been achieved as far as deployment is concerned. Many reasons are responsible for this status. Hence, the Internet is likely to be organized with both unicast and multicast enabled networks. Therefore, it is of utmost importance to design 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 data distribution using recursive unicast trees, which allow the transparent support of unicast-only routers. Additionally, HBH is original because its tree construction algorithm takes into account the unicast routing asymmetries. As most multicast routing protocols rely on the unicast infrastructure, these asymmetries impact the structure of the multicast trees. We show through simulation that HBH outperforms other multicast routing protocols in terms of the delay experienced by the receivers and the bandwidth consumption of the multicast trees.

1 Introduction

IP multicast has been addressed since more than a decade but very little has been achieved as far as deployment is concerned. Many reasons are responsible for this status. The IP Multicast architecture is composed of a service model that defines a group as an open conversation from M sources to N receivers, an addressing scheme based on IP class-D addresses, and routing protocols. In IP Multicast any host can send to a multicast group and any host can join it and receive data [1]. The IP Unicast service model is also completely open, but the potential burden caused by unauthorized senders is amplified by the group size in multicast.

The IP Multicast architecture is completed by group addressing and routing protocols. A multicast group is identified by a class-D IP address which is not related to any topological information, as opposed to the hierarchical unicast addressing model. Therefore, multicast address allocation is complicated and multicast forwarding state is difficult to aggregate. Currently, there is no scalable solution to inter-domain multicast routing.

Nevertheless, ISPs could be interested in multicast to face the increasing demand for network resources and content distribution. As a consequence, the Internet is likely to be organized with both unicast and multicast enabled networks. Therefore, it is of utmost importance to design protocols that allow the progressive deployment of the multicast service by supporting unicast clouds.

Different solutions that simplify the multicast service by reducing the distribution model were proposed [2]. EXPRESS [3] restricts the multicast conversation to *1 to N* (the *channel* abstraction), simplifying address allocation and data distribution, and still covering most of the current multicast applications. Nevertheless, source-specific multicast is not a solution to the *progressive* deployment of multicast.

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

Currently, the only alternative is to use tunnels to go through unicast-only networks. The ability to transparently support unicast routers is the main motivation of the Hop-By-Hop multicast routing protocol (HBH) we propose in this paper. HBH implements multicast distribution through recursive unicast trees, approach originally proposed in REUNITE [4]. REUNITE does not use class-D IP addresses for group identification, completely abandoning the IP Multicast addressing model.

HBH uses the unicast infrastructure to do packet forwarding with smaller routing tables, just as REUNITE does, but uses EXPRESS' channel abstraction 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 [5, 6, 7, 8]. Consequently, HBH potentially provides best routes in asymmetric networks and is suitable for an eventual implementation of Quality of Service (QoS) based routing. Additionally, HBH has a tree management algorithm that provides enhanced tree stability in the presence of group dynamics and potentially reduces tree bandwidth consumption in asymmetric networks.

This paper is organized as follows: Section 2 presents the related work, motivations and basic ideas of HBH, Section 3 describes the HBH protocol functioning and Section 4 presents a performance comparison of HBH and other multicast protocols through simulation. Section 5 concludes the paper.

2 The Basic Principles of Hop-By-Hop Multicast

This section presents previous work related to this paper, namely the EXPRESS and REUNITE protocols, and then introduce the basic principles of HBH and the problems caused by asymmetric unicast routing that motivated the design of HBH.

2.1 Related Work

EXPRESS [3] provides a simple solution to the multicast address allocation problem, introducing the channel abstraction that reduces the multicast conversation from *M to N* 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 a unicast address with a class-D address solves the address allocation problem since the unicast address is unique. The channel model also simplifies group management issues such as sender access control, although its implementation (PIM-SSM [9]) adds no group management support.

REUNITE (REcursive UNIcast TrEes) [4] implements multicast distribution based on the unicast routing infrastructure. REUNITE's basic motivation is that in typical multicast trees, the majority of 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 in all routers of the multicast tree. Therefore the idea is to separate multicast routing information in two tables: a Multicast Control Table (MCT) that is 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 which are used to recursively create packet copies as to reach all group members.

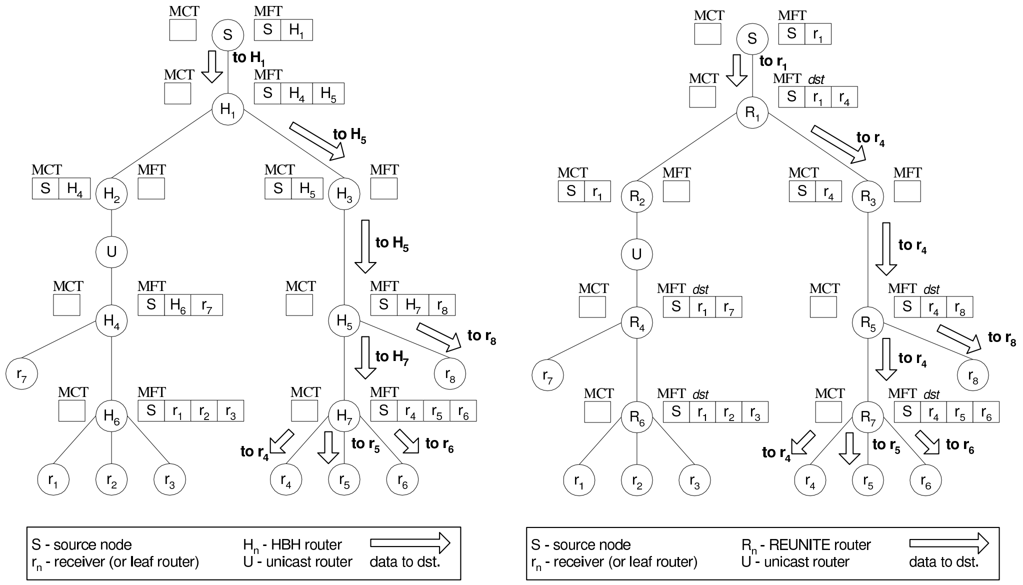
REUNITE identifies a conversation by a $<S, P>$ tuple, where S is the unicast address of the source and P is a port number allocated by the source. Class-D IP addresses are not used. As receivers join the group REUNITE populates its tables to construct the distribution tree. REUNITE uses two message types: *join* and *tree*. *Join* messages travel upstream from the receivers to the source, as *tree* messages are periodically multicast by the source to refresh soft-state of the tree. Only the branching nodes for the group $<S_1, P_1>$ keep $<S_1, P_1>$ entries in their MFT. The control table, MCT, is not used for packet forwarding. Non-branching routers in the $<S_1, P_1>$ tree have MCT entries for $<S_1, P_1>$ but no MFT entry.

2.2 Multicast distribution through recursive unicast

The basic idea of the recursive unicast approach is that packets have *unicast* destination addresses. The routers that act as branching nodes for a specific multicast group are responsible of creating packet copies with *modified* destination address in such a way that all group members receive the information.

Figure 1(a) shows how the recursive unicast data distribution works for HBH. 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.

Figure 1(b) gives an example of the recursive unicast data distribution in REUNITE. The source sends data in unicast to the first receiver that joined the group. At a branching node, R_B, incoming 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 of each packet copy is set to the receiver's unicast address). The original packet is also forwarded to r_i. In the example, S produces data packets addressed to r_1 (these packets reach r_1 unchanged). R_1 creates one packet 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 packet copy to r_8 and finally R_7 creates copies to r_5 and r_6.



(a) HBH tree.

(b) REUNITE tree.

Figure 1: Data distribution in the recursive unicast approach.

The recursive unicast technique allows the progressive deployment of the multicast service because data forwarding is based on unicast addresses. Unicast-only routers in the distribution tree are transparently supported. These routers are unable to be branching nodes of the tree but are still able to forward data packets since unicast destination addresses are used.

2.3 The risks of asymmetric routing

Asymmetric routing means that the unicast path from A to B may differ from the path from B to A. In the Internet, it may be due to different reasons. The simplest case is that of asymmetric or unidirectional links (e.g., ADSL lines or satellite links). There are also less obvious sources of asymmetric routes: routing misconfiguration and routes intentionally configured asymmetric. This kind of configuration, known as "hot-potato routing", is caused by the wish of network providers to minimize their utilization as a transit network.

Asymmetric unicast routing affects multicast routing since the majority of multicast routing protocols construct *Reverse* Shortest-Path Trees [5, 6, 7]. Data packets from the source to a receiver follow the unicast route used to go from the receiver to the source. If these paths have different characteristics, e.g. different delays, the use of the reverse SPT may be problematic to QoS deployment. The ability to construct Shortest-Path Trees is therefore advantageous for a multicast routing protocol.

REUNITE differs from previous routing protocols because it potentially constructs SPTs. (MOSPF - Multicast Open Shortest Path First [10] is the only Internet protocol that constructs SPTs.) This is possible because the *tree* messages that travel from the source to the destination nodes install forwarding state and not the *join* messages that follow the inverse direction. Nevertheless, REUNITE may fail to construct shortest-path branches in the presence of unicast routing asymmetries. A second undesirable behavior of REUNITE is that the route for one receiver may change after the departure of another receiver. This is undesirable if some QoS mechanism is to be implemented.

Figure 2 illustrates the tree construction mechanism of REUNITE with 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 multicast 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 in the direction of 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 $tree(S, r_2)$ messages downstream (upon the reception of $tree(S, r_1)$). We say that r_2 joined the channel at R_3. Data packets sent to the group (addressed to r_1) are duplicated at R_3 and addressed to r_2. Subsequent *join* messages sent by r_1 and r_2 refresh the MFT entries at S and R_3 respectively.

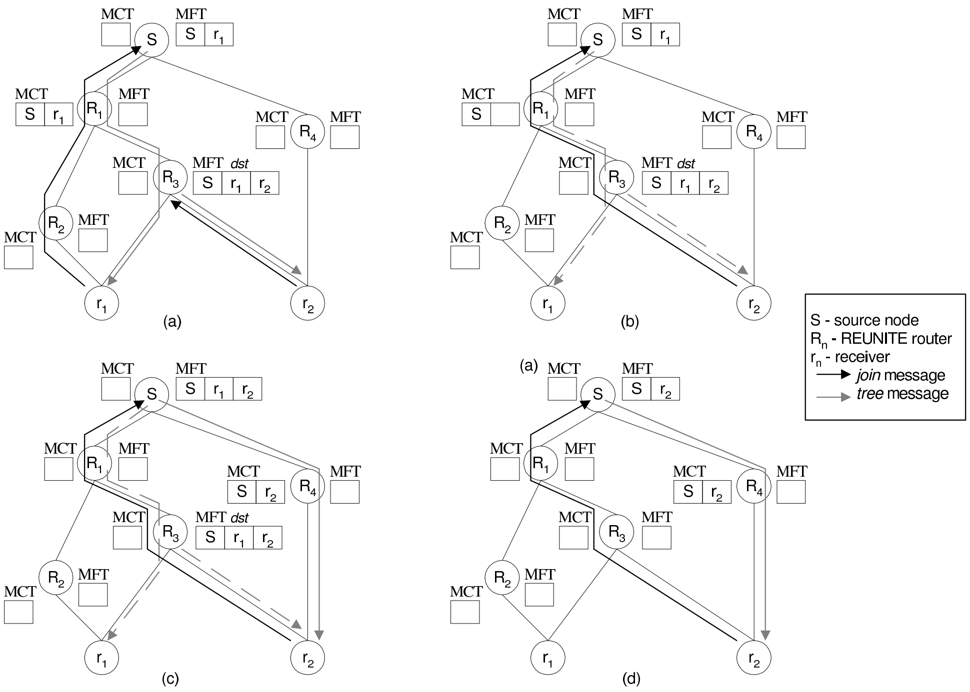


Figure 2: REUNITE's tree construction mechanism.

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)).

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

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 $<S>$ (in Figure 2, $join(S,r_1)$ refreshes the r_1 entry in S's MFT and $join(S,r_2)$ refreshes the r_2 entry in R_3's MFT).

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* times out resulting in the deletion of r_1 from S's and R_3's MFTs. As R_3 stops receiving *tree* messages, its MFT$<S>$ is destroyed (Figure 2(d)). Now, r_2 receives data through the shortest-path from S.

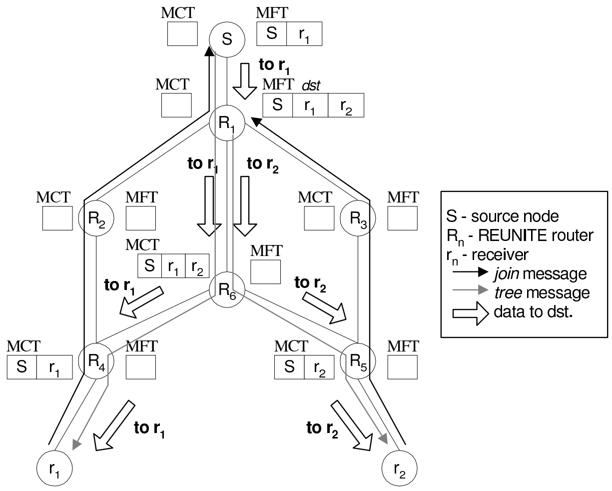


Figure 3: Packet duplication due to asymmetric routes in REUNITE.

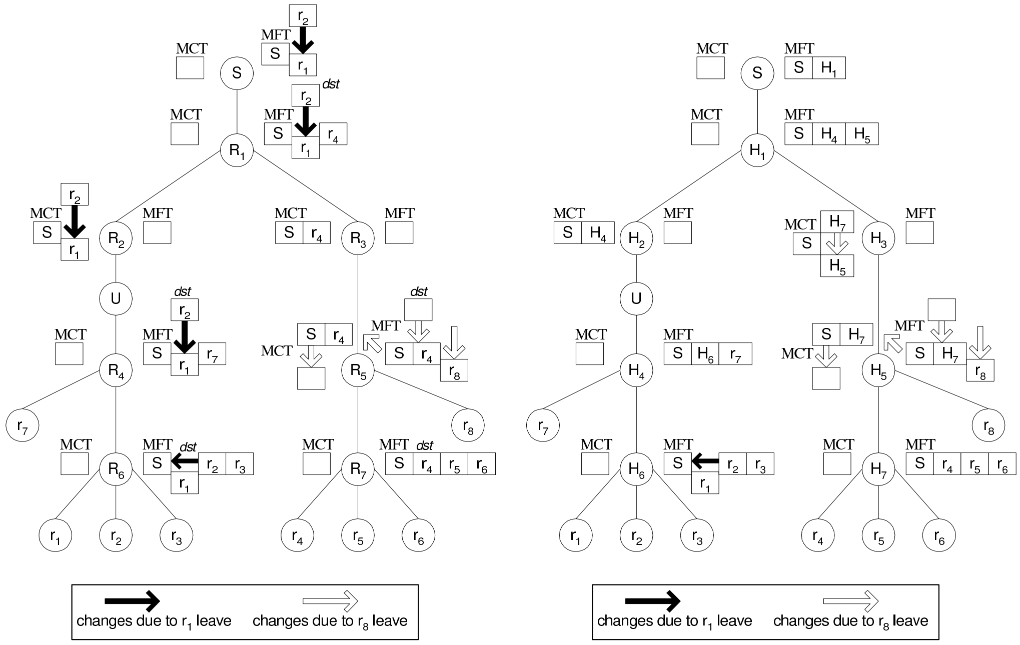
Asymmetric routing may also lead REUNITE to unneeded packet duplications on certain links.[2] Figure 3 gives an example. The first receiver, r_1, sends a $join(S,r_1)$ that follows the path $r_1{>}R_4{>}R_2{>}R_1{>}S$. The $tree(S,r_1)$ messages follow the route $S{>}R_1{>}R_6{>}R_4{>}r_1$. Suppose now that r_2 joins and that $join(S,r_2)$ follows $r_2{>}R_5{>}R_3{>}R_1{>}S$. 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 (the number of packet copies in the network) of a REUNITE tree may be larger than that of a source tree constructed by a classic protocol as PIM-SM (Protocol Independent Multicast - Sparse Mode)[7], since the RPF (Reverse Path Forwarding) algorithm ensures one unique packet copy over each network link. The next section describes HBH protocol functioning and how its tree construction mechanism is able to cope with the problems due to asymmetric unicast routing.

[2] In fact, this possibility also exists when the network has pure unicast routers or when the REUNITE router is overloaded. In both cases, the branching node must migrate to another router and may cause packet duplications in some links. For a more detailed description, the reader is referred to [4].

3 Hop-By-Hop Multicast Protocol

HBH has a tree construction algorithm that is able to better treat the pathological cases due to asymmetric unicast routes. 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 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 (in REUNITE data is addressed to MFT.<*dst*>). This choice turns the tree structure more stable than in REUNITE. A multicast channel in HBH is identified by $<S, G>$, where S is the unicast address of the source and G is a class-D IP address allocated by the source. This definition solves the address allocation problem while being compatible with IP Multicast. Therefore HBH can support IP Multicast clouds as leaves of the distribution tree.



(a) Tree reconfiguration in REUNITE. (b) Tree reconfiguration in HBH.

Figure 4: Comparison of tree reconfiguration after member departure.

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. For example, the departure of r_1 in Figure 4 has a greater impact in the tree structure of REUNITE than in that of HBH. In the worst case, HBH may need one more change than REUNITE (it happens when a branching nodes becomes a non-branching one, e.g. after the departure of r_8). In this example routes are symmetric so there is no route changes for other members when a member leaves the group. Nevertheless, tree reconfiguration in REUNITE may cause route changes to the remaining receivers, as for r_2 in the example of Figure 2. This is avoided in HBH.

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. The Appendix A gives a detailed description of the message processing rules of HBH. The basic ideas are: the first *join* issued by a receiver is never intercepted, reaching the source; the *tree* messages are periodically multicast by the source; these are combined with *fusion* messages sent by potential branching nodes to construct and refine the tree structure.

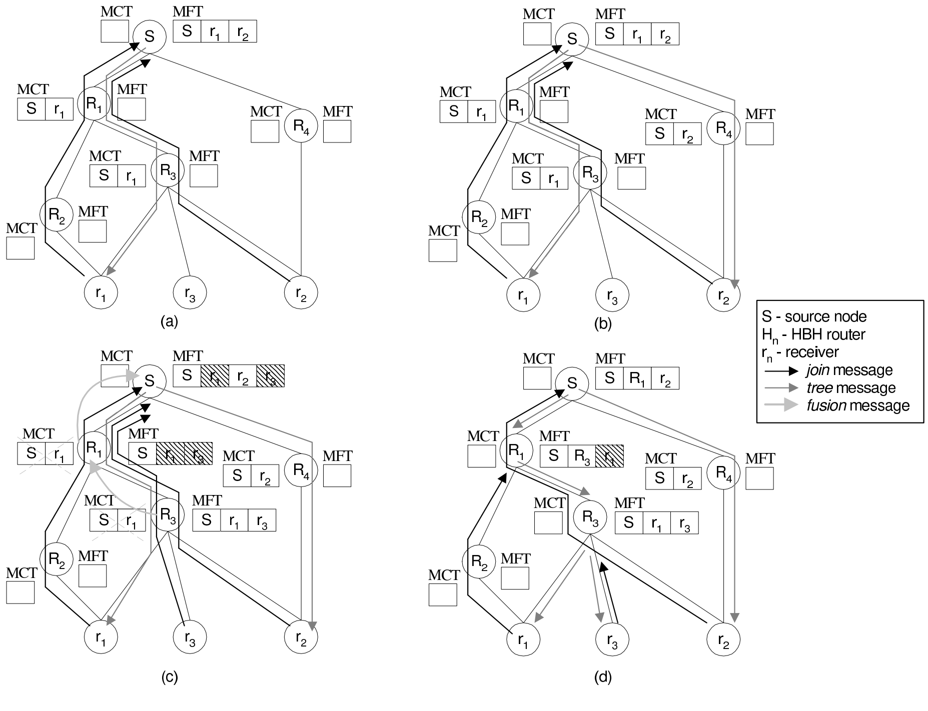


Figure 5: HBH's tree construction mechanism.

We come back to the first example of Section 2.3 to show how the tree management of HBH works. 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. Subsequent $join(S, r_1)$ 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)[11] 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 copies of the same packet that are transmitted to reach all receivers.

4.1 Simulation scenario

The first topology used in our simulations is shown in Figure 6. This topology is typical of a large ISP's network [12]. 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 [13] 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. Nodes 0 to 17 in Figure 6 are routers whereas nodes 18 to 35 are potential receivers of the multicast channel. We have also simulated a random-generated topology with 50 nodes and higher connectivity (8.6 versus 3.3).

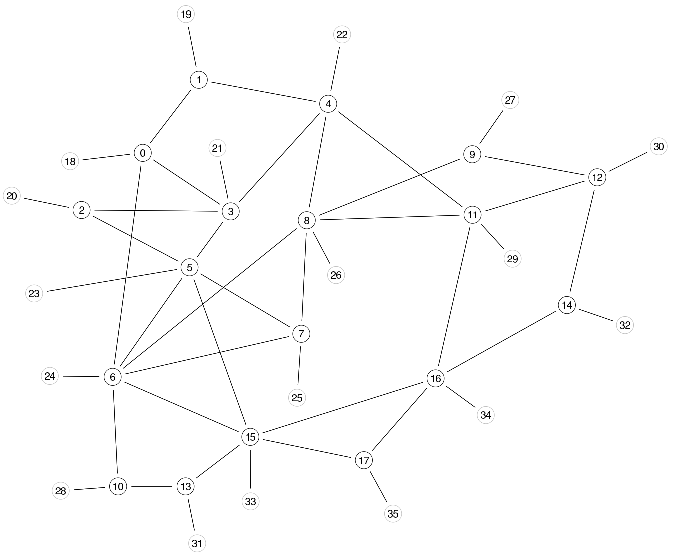


Figure 6: The ISP topology.

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 group from 1 to N where node 18 is fixed as source. A variable number of randomly chosen receivers join the channel. For each group size we realized 500 simulation runs per protocol. The plotted results are the average of the 500 experiments.

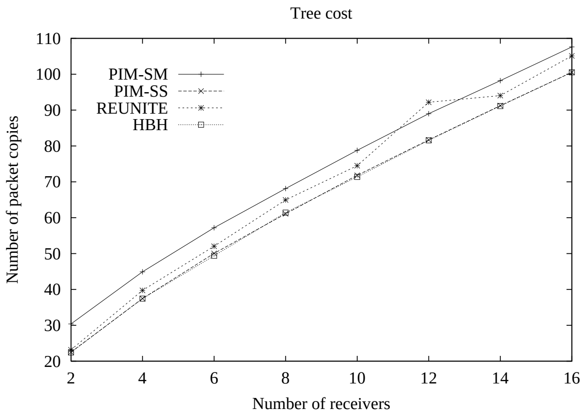
4.2 Results

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 [9], i.e., a reverse SPT. In addition to HBH, we implemented REUNITE according to [4]. All routers implement the multicast service in our experiments.

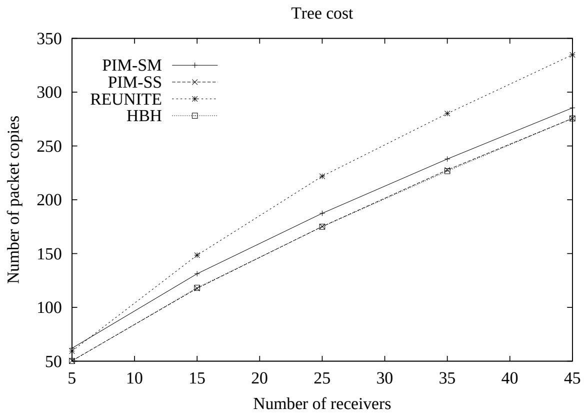
4.2.1 Tree cost

We first evaluated the cost of the trees constructed by the different multicast routing protocols. We define the cost of a tree as the number of copies of the same packet that are transmitted in the network links. Therefore, the tree cost is different from the number of links in the tree since the recursive unicast technique may send more than one copy of the same packet over a specific link. This may be due to the network's routing asymmetries (as shown is Section 2.3) but also to unicast routers inside the network that are not able to be branching nodes. Nevertheless, as in our experiments all routers are multicast capable, extra packet copies are always due to routing asymmetries.

Figure 7 shows the average cost of the multicast trees constructed by the different protocols as the number of receivers varies. For the ISP topology, PIM-SM constructs the trees with the highest cost in most cases. This result was 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). With a high probability this tree has a higher cost than the equivalent source tree. HBH and PIM-SS construct the cheapest trees. This result is expected since PIM-SS constructs source trees based on the RPF algorithm. On the one hand it guarantees that at the maximum one copy of the same packet is transmitted at each link and on the other hand that each receiver is connected to the source trough the *reverse* shortest-path. HBH has a performance similar to PIM-SS because in HBH each receiver is connected to the source through the shortest-path. Using the shortest path from source to receiver or from receiver to source does not influence tree cost.



(a) ISP topology.



(b) 50-node random topology.

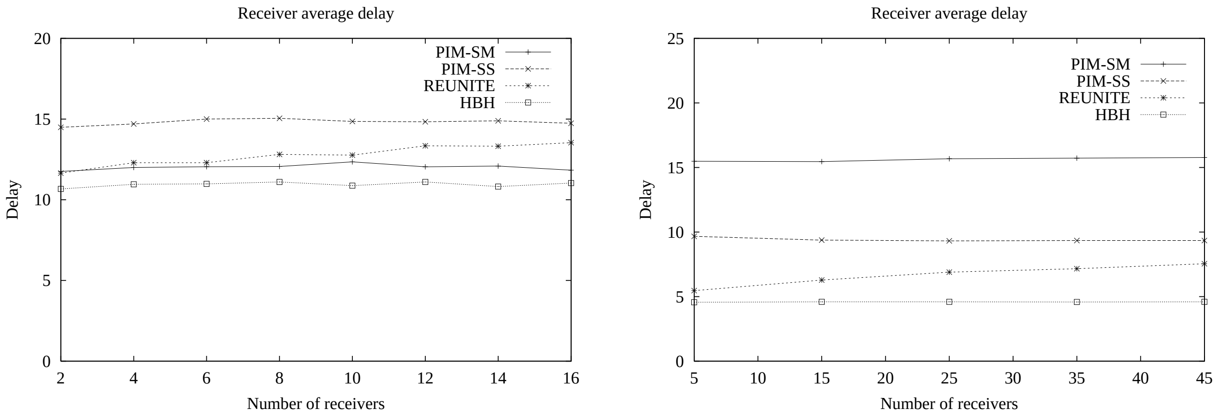
Figure 7: Average number of copies of the same packet.

The REUNITE curves in Figure 7 show that REUNITE's tree construction mechanism effectively suffers from the pathological cases produced by asymmetric unicast routing, as we presented in Section 2.3. 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. For the ISP topology, the problem is also less severe when the number of receivers is huge (receiver distribution is dense) since a big percentage of network links is anyway used in the distribution tree. Nevertheless, this is not the case for the 50-node topology. This topology has a much higher connectivity, which means that a smaller percentage of network links is used. In this topology HBH's advantage increases with the group size. REUNITE also performs worse than PIM-SM shared trees as a consequence of badly placed branching nodes which lead to useless packet duplications. The analysis of the HBH curve shows the enhanced efficiency of HBH's tree construction mechanism. In terms of tree cost, the advantage of HBH over REUNITE is as large as 5% for the ISP topology and 18% for the 50-node topology, in average over all group sizes. We conclude that HBH provides a better bandwidth utilization than REUNITE.

4.2.2 Delay

Figure 8 presents the average delay experienced by the receivers in the multicast channel for the same set of simulations. The curves show that HBH is effectively able to generate better quality routes than REUNITE in the presence of asymmetric unicast routing.

Figure 8(a) shows two unexpected results. First, PIM-SM performs better that PIM-SS in terms of delay for the ISP topology (in other words, the shared trees have a better delay performance than the source trees). This fact is explained because PIM-SS tree is a *reverse* SPT and not a SPT. So delay is not minimized. Delay is not minimized either in the shared tree constructed by PIM-SM. But in the shared tree, the paths from the source to the different receivers all have one common portion, namely, the path between the source and the RP. As data is encapsulated in unicast between the source and the RP, delay is minimized between the source and the RP. Consequently, 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 since it is a *reverse* shortest path. This explains the advantage of PIM-SM over PIM-SS. The same was not observed for the 50-node topology because this topology is larger and has a higher connectivity. Therefore going through the RP is likely to result in a longer path than going directly from the source to the receiver. The second important remark for the ISP topology is that the effect of the network asymmetries in the quality of REUNITE trees may be strong, as REUNITE performed worse than PIM-SM when the receiver set is large.



(a) ISP topology.

(b) 50-node random topology.

Figure 8: Average delay experienced by the receivers.

The delay performance of HBH is better than that of REUNITE for all group sizes, in both topologies. The advantage becomes larger as the number of receivers grows, being of 14% in average for the ISP topology. The absolute values for the 50-node topology are smaller as this topology has a higher connectivity. On the other hand, the advantage obtained by HBH over REUNITE for this topology is larger (30% in average). This is a consequence of its richer connectivity, as the routing protocol has more choices to construct the distribution tree, and is consequently more vulnerable to unicast routing asymmetries.

5 Conclusions

We presented HBH, a multicast routing protocol that implements multicast distribution through recursive unicast trees, idea originally proposed in REUNITE [4]. HBH allows the incremental deployment of the multicast service as unicast routers inside the network are transparently supported. The observation of the strengths and weaknesses of REUNITE and EXPRESS directed the design of HBH. The objectives of HBH are:

- go through unicast clouds;
- member departure should have minimum impact on the tree structure;
- provide lower cost trees in the cases where REUNITE tree construction fails;
- guarantee that members receive data 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 in the presence of asymmetric unicast routing. HBH also provides a better network utilization as it is able to construct recursive unicast trees minimizing packet duplication. This is an advantage if the network bandwidth is scarce. Additionally, HBH's delay performance makes it a routing protocol adapted to applications that do not support large delays such as interactive ones.

The results obtained through simulation show that the unicast routing asymmetries affect the performance of the multicast routing protocol. Additionally, HBH is a promising approach as its tree construction algorithm outperformed REUNITE in terms of the tree cost and the delay experienced by the receivers. The advantage of HBH grows with larger and more connected networks. Our future work includes the formal definition of the interface between HBH and classical IP Multicast, and to study the possibility of including QoS parameters inside HBH's tree construction algorithm.

Acknowledgments

We would like to express our gratitude to Supratik Bhattacharyya (Sprint Labs.), Mark Handley (ACIRI), Luigi Rizzo (Univ. of Pisa), and Lorenzo Vicisano (Cisco Systems) for their constructive comments on this work.

References

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

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

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

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

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

[6] 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*, vol. 4, no. 2, pp. 153–162, Apr. 1996.

[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] C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint communication: A survey of protocols, functions and mechanisms," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 3, pp. 277–290, Apr. 1997.

[9] S. Bhattacharyya, C. Diot, L. Giuliano, R. Rockell, J. Meylor, D. Meyer, and G. Shepherd, *A Framework for Source-Specific IP Multicast Deployment*, July 2000. Internet-draft: draft-bhattach-pim-ssm-00.txt.

[10] J. Moy, *Multicast Extensions to OSPF*. RFC 1584, Mar. 1994.

[11] 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.

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

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

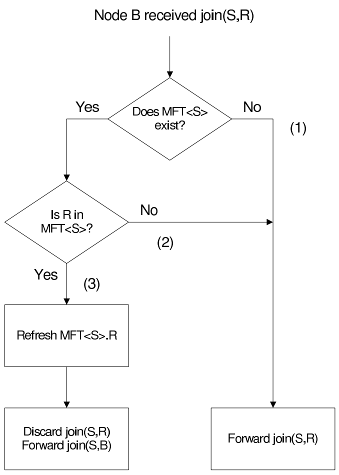
A Message Processing Rules of HBH

Figure 9 presents the processing rules of the three message types. Each receiver r periodically sends a $join(S,r)$ in unicast to the source. The source periodically multicasts a $tree$ message for each $<S,G>$ channel.

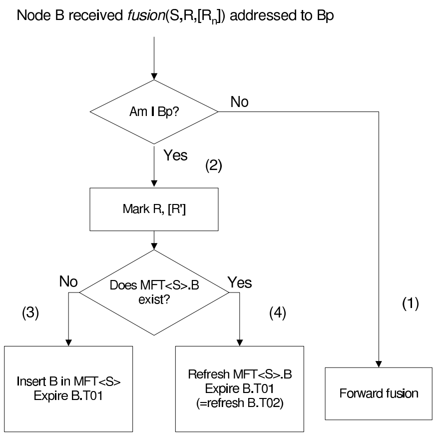
Join **message** (Figure 9(a)) - When router B receives a $join(S,R)$, it should forward this message unchanged if B has no MFT (1) or if R is not in B's MFT (2). Only if B's MFT has a R entry, B intercepts the $join(S,R)$ and sends a $join(S,B)$ afterwards. It means that B is a branching node of the channel $<S,G>$ (3).

Tree **message** (Figure 9(c)) - A $tree(S,R)$ message received by router B is treated and forwarded in multicast. If B is a branching node, it may receive a $tree$ message addressed to B. In this case B discards the message and sends a $tree$ message to each node present in its MFT (1). If B is a branching node and $tree(S,R)$ is not addressed to B, there is two possibilities: R is a new receiver (in which case B inserts R in its MFT and sends a $fusion$ message upstream) (2) or R is present in B's MFT - which means that B does not receive $join(S,R)$ messages from R - and in this case B simply has to refresh the R entry in its MFT and to send a $fusion$ message upstream (3). If B is not a branching node, there is two possible cases: B was not in S's distribution tree in which case B creates a MCT containing R (4), or B was already in the distribution tree but was not a branching node (in which case B has a MCT$<S>$) (5). If R was already present in B's MCT, nothing has changed and B simply refreshes its MCT (6). If R is not present in the MCT, then maybe the MCT is stale in which case R replaces the previous MCT entry, or the MCT is up to date which means that there is a second receiver that will receive data through B, so B becomes a branching node. This implies the creation of a MFT, the destruction of the MCT, and a $fusion$ message to be sent upstream (8). The $fusion$ messages produced by B contain all the nodes that B maintains in its MFT - the nodes for which B is branching node.

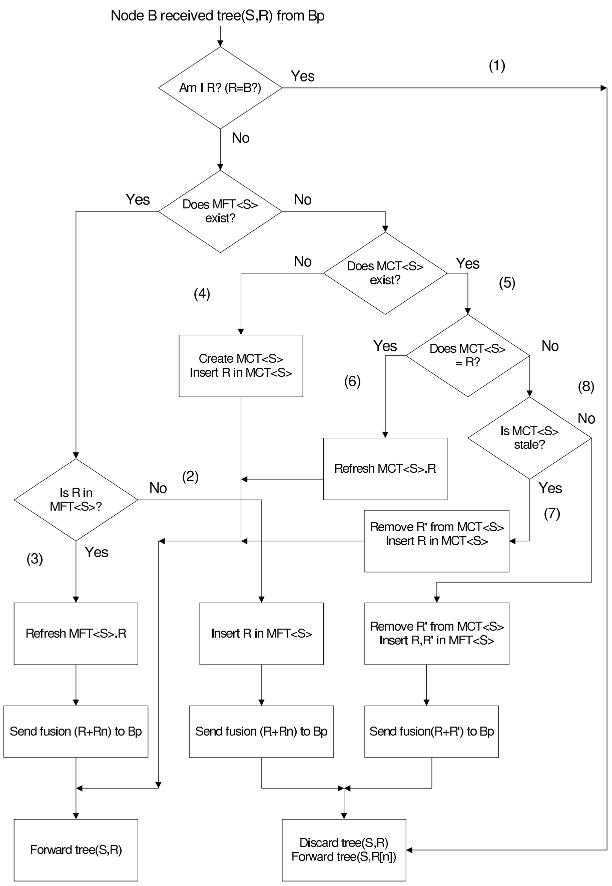
Fusion **message** (Figure 9(b)) - Suppose router B receives a $fusion(S,R,...R_n)$ from node B_p. If the message is not addressed to B, then B simply forwards it upstream (1). If the message is addressed to B, then B marks the receiver entries in its MFT that are listed in the $fusion$ message (2). A marked entry in the MFT is used to $tree$ message forwarding but not for data distribution. B_p is added to B's MFT if it was not previously present. In addition, B_p's *t1* timer is expired - B_p becomes stale (3). Consequently, B_p will be used for data forwarding, but not for $tree$ message forwarding. If B_p was already present in B's MFT, then B_p's *t2* timer is refreshed (it avoids the destruction of the B_p entry),



(a) *Join* message.



(b) *Fusion* message.



(c) *Tree* message.

Figure 9: Message processing in HBH.

but its *t1* timer is kept expired (4).

If afterwards B_p (the node that produced the $fusion$ for $R, ...R_n$) receives the $join$ messages produced by any of $R, ...R_n$, it intercepts them and send a $join(S, B_p)$ upstream. In this case the $R, ...R_n$ entries in B's MFT will eventually timeout and be destroyed, while the B_p entry in B's MFT is refreshed by the $join(S, B_p)$. If B_p does not receive $join$ messages from $R, ...R_n$, the emission of $fusion$ messages continues since there is another node upper in the tree that receives the $join(S, R, ...R_n)$ and periodically produce the $tree(S, R, ...R_n)$ messages to these receivers. Nevertheless, this node will not forward data to these receivers, but to B_p instead since the receivers' entries are marked. B_p is responsible for data duplication. It is in this way that HBH is able to manage the second asymmetry problem presented in Section 2.3.