Analyzing the Effects of Asymmetric Unicast Routes on Multicast Routing Protocols

Luís Henrique M. K. Costa[1,2], Serge Fdida[1], and Otto Carlos M. B. Duarte[2]

[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

[2]Grupo de Teleinformática e Automação
Universidade Federal do Rio de Janeiro
P.O. Box 68504 – 21945-970 – Rio de Janeiro – Brasil
otto@gta.ufrj.br

Abstract

Different multicast routing protocols construct their distribution trees based on the information obtained from the unicast routing infrastructure. Nevertheless, the design of most of these protocols do not take into account that unicast routes may be asymmetric. Indeed, unicast routes in the Internet are very asymmetric. The effects on the quality of the multicast trees vary according to the routing protocol. These are particularly important for protocols that use the recursive unicast approach to allow the progressive deployment of the multicast service. This paper analyses the effects of asymmetric unicast routing on different multicast protocols. We concentrate on two approaches that implement the multicast service trough recursive unicast trees, HBH (Hop-By-Hop multicast routing protocol) and REUNITE (Recursive UNIcast TrEes). Both protocols construct source-specific trees exclusively, which simplify address allocation. As data packets have unicast destination addresses, pure unicast routers are transparently supported. The branching-nodes recursively create packet copies to implement the distribution. Nevertheless, the tree construction algorithms implemented by HBH and REUNITE are different. The design of HBH takes into account the unicast routing asymmetries of the network. HBH is able to always construct a Shortest-Path Tree. Consequently, HBH provides shorter delay routes in asymmetric networks, and provides smaller bandwidth consumption because useless data duplication is avoided. The results obtained from simulation show the effects of unicast routing asymmetries in the different multicast protocols.

Keywords: *multicast, routing, service deployment*

1 Introduction

The IP Multicast [1] architecture is composed of a service model that defines a group as an open conversation between *M* sources and *N* receivers and routing protocols that implements a distribution tree on top of the network infrastructure. In spite of a decade of research, IP Multicast was not yet widely deployed. This is mainly due to the complexity of the protocol architecture, additionally to the lack of a "killer" multicast application. As a consequence, the research community started to think about alternative models to multicast distribution, including application layer approaches.

Some proposals were made to simplify the multicast service [2]. EXPRESS [3] first proposed the restriction of the multicast conversation to a single source, which largely simplified the service implementation. The channel abstraction introduced in EXPRESS was adopted in PIM-SSM (Protocol Independent Multicast - Source-Specific Multicast) [4] which is currently at the end of its standardization process. REUNITE [5] implements multicast distribution through recursive unicast trees, being able to support pure unicast routers. Both protocols are discussed in more detail in Section 2.

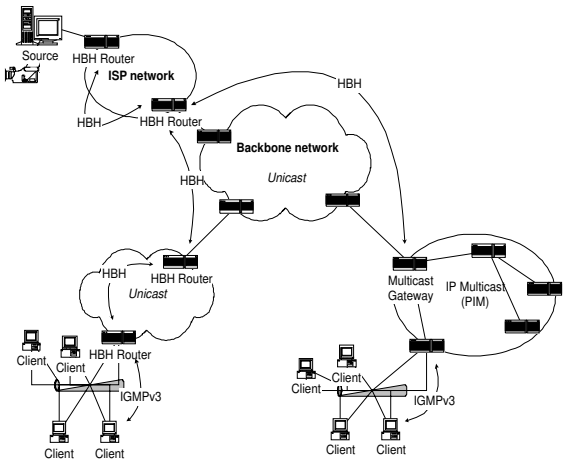


Figure 1 : Multicast service deployment scenario.

The analysis of these works lead us to the proposition of the Hop-By-Hop multicast routing protocol (HBH) [6]. HBH uses the unicast infrastructure to do packet forwarding with smaller routing tables, just as REUNITE does, but uses the channel abstraction of PIM-SSM (a *(S,G)* pair) to keep compatibility with IP Multicast. Figure 1 illustrates the multicast service deployment scenario that directed the design of HBH. Version 3 of IGMP (Internet Group Management Protocol) [7] is used due to its source filtering capability. HBH constructs Shortest-Path Trees (SPT) instead of Reverse SPTs as most routing protocols do [8,9,10,11].

Various multicast routing protocols construct their distribution trees based on the information obtained from the unicast routing infrastructure. Nevertheless, the design of most of these protocols do not take into account that unicast routes may be asymmetric. Indeed, in the case of the Internet, unicast routes are very asymmetric [12]. Nevertheless, the literature lacks an investigation on the effects of these asymmetries on the quality of the multicast routing trees.

The impact of asymmetric unicast routes may be especially important for multicast routing protocols that use recursive unicast trees to realize the multicast distribution, such as REUNITE and HBH. These protocols may be lead to produce a higher number of data packets as well as larger delay for the receivers if unicast routes are asymmetric. Additionally, other IP Multicast routing protocols that construct Reverse SPTs are also obviously subject to the second effect. In this paper we evaluate the effects of unicast routing asymmetries on the two types of multicast routing protocol.

The paper is organized as follows: Section 2 briefly presents the routing protocols studied, Section 3 illustrates the different effects of asymmetric unicast routing on the construction of multicast trees, and Section 4 compares the different protocols through simulation. Section 5 concludes the paper.

2 New Multicast Routing Protocols

2.1 PIM-SSM

PIM-SSM (Protocol Independent Multicast - Source-Specific Multicast) [4] incorporated a simple solution to the multicast addressing problem, the channel abstraction, which 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 the address allocation provides an unique identifier. This service model also simplifies group management issues such as sender access control, although the protocol specification incorporates no group management support.

2.2 REUNITE

REUNITE (REcursive UNIcast TrEes) [5] implements multicast distribution based on the unicast routing infrastructure. The basic observation is that in typical Internet multicast trees, most routers are simple relay nodes that forward packets from one incoming interface to one outgoing interface. 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 tree structure. 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 $<S>$.

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, incoming data packets are addressed to the first receiver, r_i, that joined the group in the sub-tree below R_B. Receiver r_i is stored in a special MFT entry, MFT$<S>$.*dst* (In the rest of the paper, we interchangeably use $<S>$ and $<S,P>$ to refer to the multicast channel.). The branching router 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.

REUNITE can be progressively deployed, because the use of unicast destination addresses turns the protocol able to support unicast routers in the distribution tree. These routers are not able to be branching nodes but are still able to forward data packets.

2.3 HBH

HBH (Hop By Hop multicast routing protocol) [6] 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* (except 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 PIM-SSM. This definition solves the multicast address allocation problem while being compatible with IP Multicast, because HBH can easily support IP Multicast clouds as leaves of the distribution tree.

HBH provides an enhanced stability of the tree structure when compared to REUNITE, because the number of table entries that have to be modified after a member's departure is minimized. This is possible because the MFT entry corresponding to a receiver is located at the branching node nearest this receiver.

HBH uses three messages in tree construction: *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 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 refine the tree structure 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$<S>$. The table 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. A detailed description of the message processing rules of HBH can be found in [6].

3 The Problems of Asymmetric Unicast Routing

Asymmetric routing in the Internet may have different
origins [12]. The simplest case is that of asymmetric or unidirectional links (e.g., ADSL lines or satellite links). There are also other sources of asymmetric routes: routing

misconfiguration and routes intentionally configured asymmetric. One such phenomenon is known as "hot-potato routing" and is caused by economical issues. Routes are configured in such a way that traffic destined outside one's network will leave it as soon as possible. For example, suppose two ISPs, **A** and **B**, that both provide connectivity through the US territory (Figure 2). Traffic generated at the East Coast in **A**'s network, and destined to a customer in the West Coast connected to **B** will be routed to **B**'s network as soon as possible, i.e. in a peering point located at the East Coast. In this way, **A** avoids using its own links to cross the country since these links are a scarce resource. On the other direction, **B** uses the same strategy causing routes between **A** and **B** to be asymmetric.

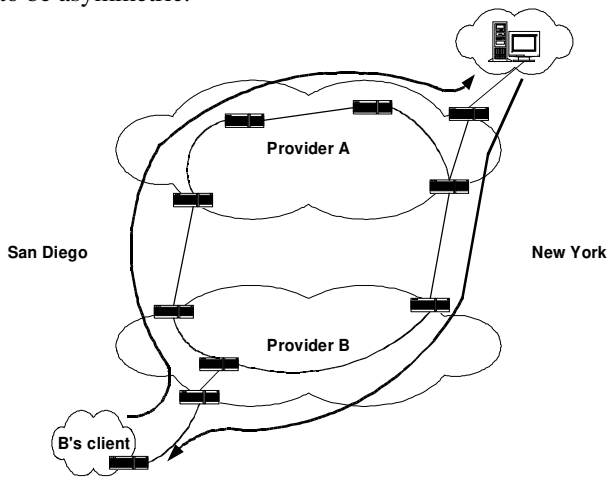


Figure 2 : Example of asymmetric routes creation.

Routing measurements in the Internet have shown that the percentage of asymmetric routes is high. The analysis in [12] evaluated about 10,000 pairs of sites. Only major routing asymmetries were considered, where the virtual paths differ by one city or AS (Autonomous System). About a half of the routes measured differed by one city or more, and about 30% of the routes were asymmetric with at least one AS of difference, which still is a high percentage.

Asymmetric unicast routes affect multicast routing since the majority of multicast routing protocols construct *Reverse* Shortest-Path Trees [8, 9, 10]. Data packets from the source to a receiver follow the unicast route used from the receiver to the source. These paths may have different characteristics, e.g. propagation delays, and thus different source-to-receiver transmission delays. Applications such as adaptive multicast video distribution [13, 14] can benefit from shortest-path trees because smaller round-trip times allow faster adaptation. The ability to construct Shortest-Path Trees is therefore advantageous to the multicast routing protocol. In the rest of this section, we illustrate different scenarios where the asymmetric unicast routing potentially causes problems in the construction of the multicast tree.

3.1 Shortest-path tree construction

One of the characteristics that differentiates REUNITE from previous routing protocols that REUNITE potentially constructs SPTs. This is due to the two message types REUNITE uses, the *tree* messages sent from the source to the destination and the *join* messages that follow the reverse path. The problem is that REUNITE may construct non shortest-path branches if unicast routing is asymmetric.

Figure 3 illustrates REUNITE tree construction mechanism and gives an example where it fails to construct a SPT. Suppose the unicast routes: $r_1 \rightarrow R_2 \rightarrow R_1 \rightarrow S$; $S \rightarrow R_1 \rightarrow R_3 \rightarrow r_1$; $r_2 \rightarrow R_3 \rightarrow R_1 \rightarrow S$; $S \rightarrow R_4 \rightarrow r_2$. Suppose the following events: receiver r_1 joins $<S,P>$, r_2 joins $<S,P>$, and r_1 leaves the group.

Receiver r_1 subscribes to the channel by sending a *join*(S,r_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. The source S then starts sending *tree*(S,r_1) messages to r_1. These *tree* messages install soft-state for $<S,P>$ in the routers traversed downstream. Routers 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. Router 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. Subsequent *join* messages sent by r_1 and r_2 refresh the MFT entries at S and R_3 respectively.

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 since R_3 intercepts *join*(S,r_2), data follows the path $S \rightarrow R_1 \rightarrow R_3 \rightarrow r_2$, the same as *tree* messages from S down to r_2 (Figure 3 (a)).

Now receiver 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 a period of time the r_1 entry becomes stale. A second timer is created and will eventually destroy the r_1 entry. As r_1 is stale, S now sends stale *tree*(S,r_1) messages (Figure 3(b)). The stale *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 stale *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,r₂)* messages are no longer intercepted by R_3 (as its MFT<*S*> is stale) and reach *S*. Receiver r_2 now joins <*S,P*> at *S* (Figure 3(c)). The entry corresponding to r_1 will eventually be deleted from *S*'s and R_3's MFTs. As R_3 stops receiving *tree* messages, its MFT is destroyed (Figure 3(d)). In this new configuration, r_2 receives data through the shortest path from *S*.

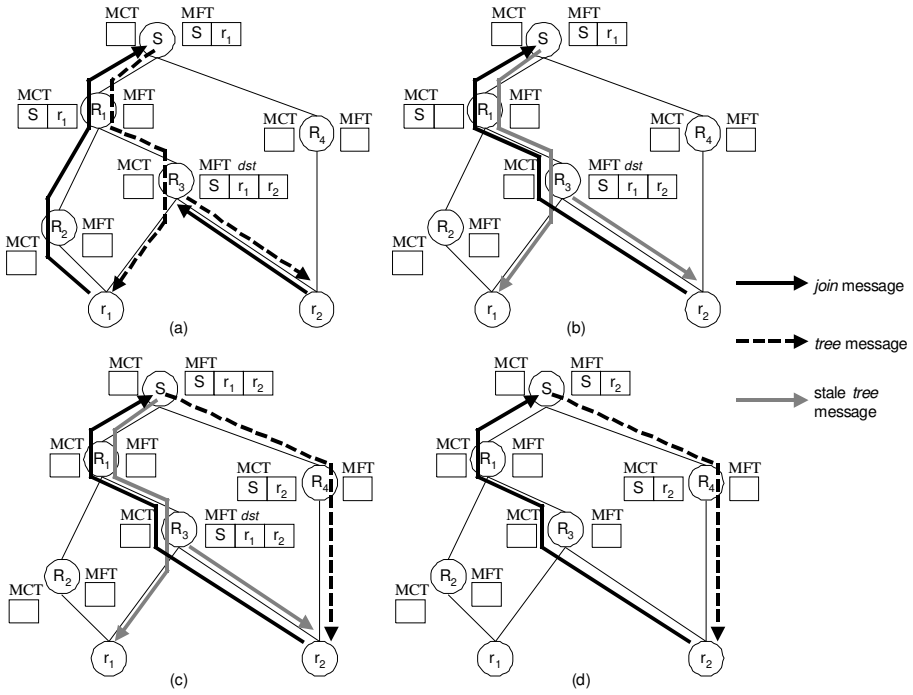


Figure 3 : REUNITE's tree construction.

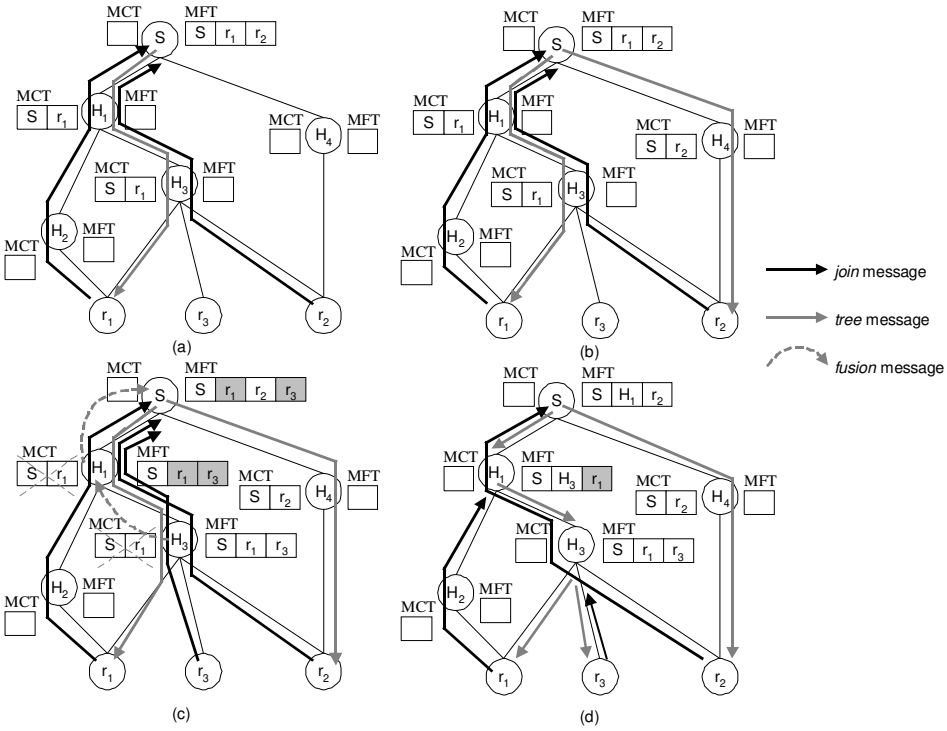


Figure 4 : HBH's tree construction.

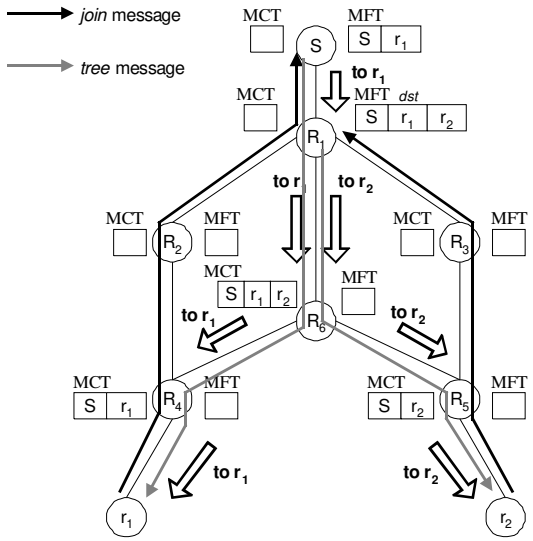
Now we consider the same scenario and HBH (Figure 4). Receiver r_1 joins the multicast channel at S which starts sending *tree(S,r1)* messages. These messages create a MCT<*S*> containing r_1 at H_1 and H_3 (Figure 4(a)). When r_2 joins the group by sending the first *join(S,r2)*, this message is not intercepted and reaches S (the first *join* message is never intercepted). The *tree(S,r2)* produced by the source create MCT<*S*> state at R_4 (Figure 4(b)). Both receivers are connected to the source through the shortest path.

Suppose now that r_3 (unicast routes: $S \rightarrow H_1 \rightarrow H_3 \rightarrow r_3$ and $r_3 \rightarrow H_3 \rightarrow H_1 \rightarrow S$) joins the channel. It sends a *join(S,r3)* to S, which starts sending *tree(S,r3)* messages. As H_1 receives two different *tree* messages, it sends a *fusion(S,r1,r3)* 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 H_1 to it. In the same way as H_1, H_3 receives *tree(S,r1)* and *tree(S,r3)* messages and thus send a *fusion(S,r1,r3)* to the source (Figure 4(c)). H_3's MFT now contains r_1 and r_3. Subsequent *join(S,r1)* messages are intercepted by H_1 and refresh the r_1 marked entry in H_1's MFT. The *join(S,r3)* messages refresh the r_3 MFT entry at H_3. S sends data addressed to H_1, that sends it addressed to H_3. H_3 sends copies to r_1 and r_3. Subsequently, as S receives no more *join(S,r1)* neither *join(S,r3)* messages, the corresponding MFT entries are destroyed. The final structure is shown in Figure 4(d). In this way, HBH is able to use the good branching point to the distribution tree.

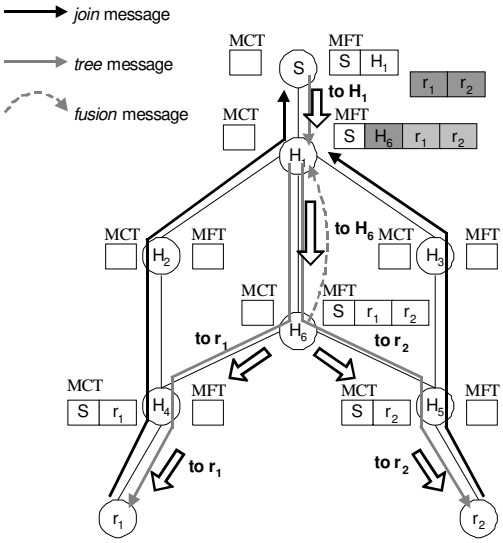
3.2 Useless packet duplication

Asymmetric routes may lead REUNITE to unneeded packet duplications.[1] Figure 5(a) gives an example. The first receiver, r_1, sends a *join(S, r1)* that follows the path $r_1 \rightarrow R_4 \rightarrow R_2 \rightarrow R_1 \rightarrow S$. The *tree(S, r1)* messages follow the route $S \rightarrow R_1 \rightarrow R_6 \rightarrow R_4 \rightarrow r_1$. Suppose now that r_2 joins and that *join(S, r2)* follows $r_2 \rightarrow R_5 \rightarrow R_3 \rightarrow R_1$. The *tree(S, r1)* (produced by S) and the *tree(S, r2)* (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 larger than that of a source tree constructed by a classic protocol as PIM-SM [10] since the RPF (Reverse Path Forwarding) algorithm ensures one unique packet copy over each network link.

[1] In fact, this possibility also exists when the network has pure unicast routers or when the REUNITE router is overloaded [5].



(a) REUNITE.



(b) HBH.

Figure 5 : Packet duplication scenario.

The *fusion* messages used by HBH cope with the problem of Figure 5. The first *join(S,r2)* will reach the source. After receiving different *tree* messages for r_1 and r_2, router H_1 sends a *fusion(S,{r1,r2})* to *S*. Subsequent *join(S,r1)* and *join(S,r2)* messages will be intercepted by H_1. At its turn, router H_6 receives two different *tree*s and sends a *fusion(S,{r1,r2})* upstream. In this case, however, it will never receive *join* messages issued by receivers r_1 and r_2. The consequence is that H_6's entry in H_1 will be kept stale and r_1 and r_2 entries will be fresh, but *marked*. Thus, data will be produced to H_6 as control will be addressed to r_1 and r_2. The design choice of HBH imposes some control overhead but minimizes data duplication.

3.3 Temporary loop creation

The first simulations executed with REUNITE showed that in some cases its algorithm leads to a huge number of control messages. The analysis of these scenarios shown a temporary loop creation. Suppose the following example (Figure 6), where the unicast routes between the source and the receivers are: $r_1 \rightarrow R_2 \rightarrow R_3 \rightarrow R_1 \rightarrow S$; $S \rightarrow R_1 \rightarrow R_2 \rightarrow r_1$; $r_2 \rightarrow R_2 \rightarrow R_1 \rightarrow S$; $S \rightarrow R_1 \rightarrow R_2 \rightarrow r_1$.

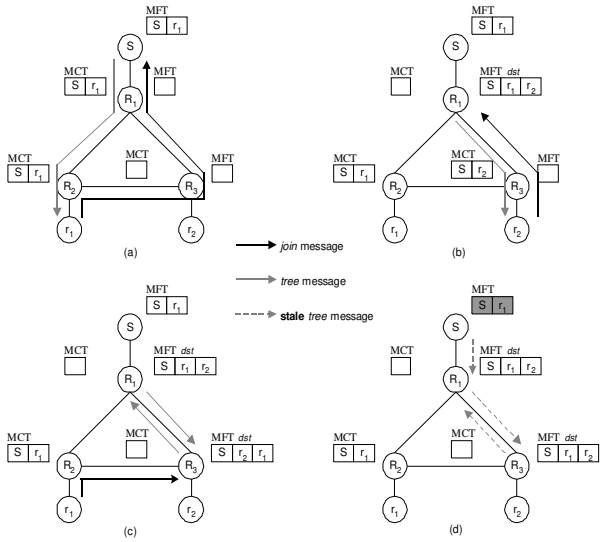


Figure 6 : Temporary loop creation in REUNITE.

First, r_1 joins the channel at the source, *S* (Figure 6(a)), with the creation of MCT state at routers R_1 and R_2. Consequently, when receiver r_2 joins the channel, R_1 intercepts the join message and installs r_2 in its MFT (Figure 6(b)). Thus, R_3 will receive *tree(S,r2)* messages produced by R_1 and create MCT<*S, r2*> state. As a consequence, when the next *join* message issued by r_1 will reach R_3, this router will create a MFT<*S*>, with r_2 as MFT<*S*>.*dst* and r_1 as a regular entry (Figure 6(c)). If the route from R_3 to r_1 traverses R_1, a temporary loop of *tree* messages is created between R_1 and R_3. This loop will last until the destruction of the MFT<*S*> state at the source, as *S* will no longer receive the *join* messages issued by r_1. One can imagine a loop detection mechanism, such as controlling the interface *tree* messages are received through (with the cost of keeping some more state). It is also possible to minimize the effects of this temporary loop. For example, if the production of *tree* messages is delayed the amount of control packets is reduced (i.e., a router does not multicast a *tree* message up to the reception of a *tree*, but only at specific intervals instead).

Now, consider this temporary loop scenario and HBH (Figure 7). The unicast routes are: $r_1 \rightarrow H_2 \rightarrow H_3$

$\rightarrow H_1 \rightarrow S$; $S \rightarrow H_1 \rightarrow H_2 \rightarrow r_1$; $r_2 \rightarrow H_3 \rightarrow H_1 \rightarrow S$; $S \rightarrow H_1 \rightarrow H_3 \rightarrow r_2$.

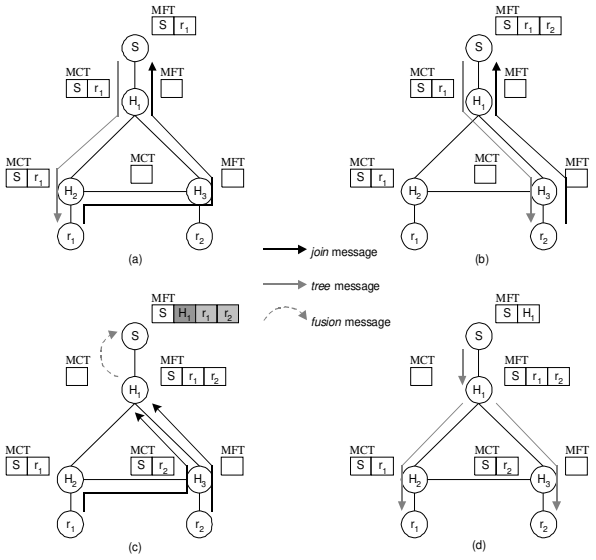


Figure 7 : Avoiding loop creation in HBH.

Suppose r_1 is the first receiver to join the channel. The source creates a MFT<S> where r_1 is installed. Routers H_1 and H_2 create tree control state upon the reception of the *tree(S,r1)* messages (Figure 7(a)). Now receiver r_2 joins the group. At this time, there is no state for r_2 in the tree, so the *join(S, r2)* reaches S, that installs r_2 in its MFT and starts to produce *tree(S, r2)* (Figure 7(b)). At the next step, H_1 will produce a *fusion(S,{r1,r2})* (Figure 7(c)). From now on, H_1 will intercept *join* messages issued by r_1 and r_2, and produce *join(S,H1)* messages. As a consequence, H_1 is the router to produce *tree(S,r1)* as well as *tree(S,r2)* messages. Figure 7(d) shows the stable tree structure.

Temporary loop creation is avoided in HBH because the creation of forwarding state is always consequence of messages that travel downstream, the *tree* messages sent by the source to the receivers. *fusion* messages can also instantiate MFT state, but as the production of *fusion* messages is itself consequence of different *tree* message reception, *and* because the "fusion" process is done hop by hop, the produced forwarding structure is always a SPT.

4 Performance Analysis

We used NS (Network Simulator) [15] to simulate the different multicast routing protocols. 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 the simulations has 18 nodes and was obtained from a large ISP's network [16]. Without loss of generality, we suppose that only one receiver is connected to each node in the topology. In other words, we did not simulate the filtering provided by IGMP [17] before the first-hop router. One node is chosen as source and a different number of routers is connected to the tree.

To simulate the asymmetric unicast routes, two costs, $c(n_1, n_2)$ and $c(n_2, n_1)$, are associated 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. The simulator 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 [10] protocol. 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 and therefore simulates the tree structure of PIM-SSM [4] (a reverse SPT). In addition to HBH, we implemented REUNITE according to [5]. All routers implement the multicast service in our experiments.

Tree cost - We define the cost of the 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 the same link. This may be due to the network's routing asymmetries (Section 3) 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.

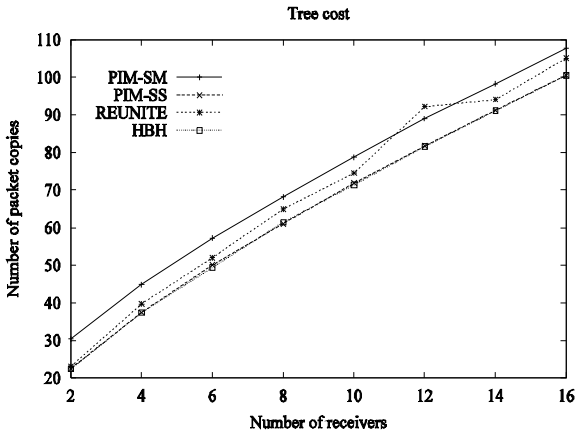


Figure 8 : Tree cost comparison.

Figure 8 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 8 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 9 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.

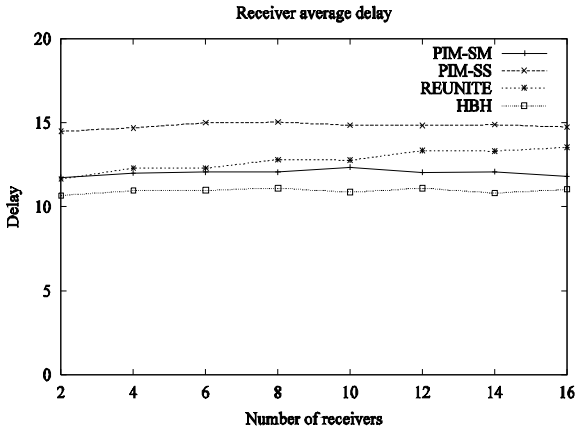
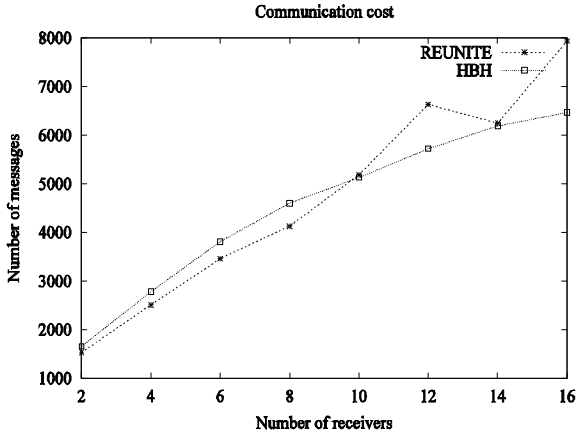


Figure 9 : Receiver delay comparison.

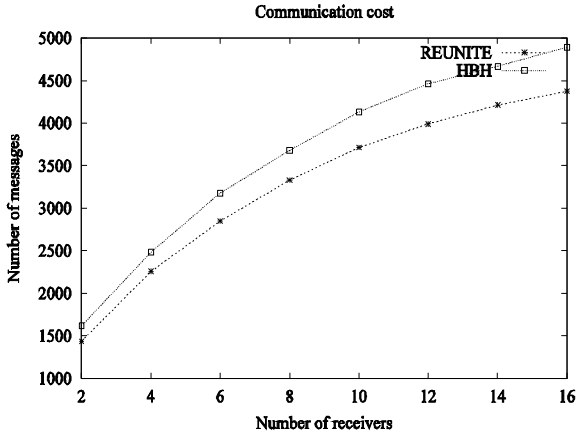
HBH is effectively able to generate better quality routes than REUNITE in the presence of asymmetric unicast routing. The performance of HBH is better than that of REUNITE for all group sizes. The advantage becomes larger as the number of receivers grows, being of 14% in average.

Message cost analysis - In this section we compare the message processing overheads of HBH and REUNITE, i.e., the amount of control messages used by the protocol to keep the multicast distribution tree. To make a fair evaluation, both protocols use the same join refresh periods (the interval between the production of consecutive *join* messages by a receiver), as well as the same period of *tree* message production and timeout values (used to "timeout" the table entries and eventually destroy them). Additionally, two sets of experiments were made. In the first one, routes are asymmetric as in all previous experiments. Nevertheless, as investigated in Section 3, REUNITE may produce temporary loops in certain scenarios. In these cases a huge number of messages is produced (thus HBH largely overperforms REUNITE) and the average processing overhead of REUNITE is increased. As one may argue that these "pathological" scenarios may be rare in the real world, as well as to have an idea of the overhead of HBH over REUNITE in the more common cases, we conducted some simulations were the routes are completely symmetric.

This is not the scenario HBH was designed for (nor is Internet's reality), but can give an idea of HBH's algorithm cost.



(a) Asymmetric routes.



(b) Symmetric routes.

Figure 10 : Average number of control messages.

These experiments were conducted with the ISP topology. Figure 10 shows the results obtained with asymmetric routes. They confirm our statements on the problems REUNITE may incur. Nevertheless, these results can not lead to other conclusions, as the amount of useless messages produced during the temporary loop is a function of the link bandwidth and propagation delay. This is because in our implementation, a *tree* message received by a router is immediately forwarded. If two routers form a "*tree*" loop, the amount of messages produced is only limited by the physical path connecting them. We did not add any of the loop avoidance mechanisms mentioned in Section 3.

Figure 10(b) shows the results obtained with symmetric routes. In this case, HBH does show a message processing cost larger than REUNITE. This is explained by HBH's more complex algorithm. To guarantee a SPT construction, the *first join* message issued by a receiver always reaches the source; additionally, HBH will in some cases continue to produce *fusion* messages (as a way to avoid useless data-packet duplication, for example router H_6 in Figure 5(b)). Figure 10(b) shows that HBH's control overhead compared to REUNITE is about 11%. If we consider that for the same topology HBH overperformed REUNITE by 5% and 14%, terms, respectively, of tree cost and delay, we may conclude that HBH's algorithm complexity is compensated. This is particularly true if we take into account that HBH's advantage applies to the *data* traffic, and that its control overhead is percentually smaller in the case of asymmetric networks, such as the Internet.

5 Conclusions

In this paper, we investigated the effects of asymmetric unicast routing on different multicast routing protocols that use the unicast routes to construct their multicast trees. More specifically, we compared three recent proposals, namely, PIM-SSM, REUNITE, and HBH. PIM-SSM is a modified version of PIM-SM that exclusively constructs source-specific trees. On the other hand, REUNITE and HBH are multicast routing protocols that implement data distribution through recursive unicast trees. These protocols allow the progressive deployment of the multicast service as unicast routers are transparently supported. Nevertheless, HBH was designed with the asymmetric unicast routes of the Internet in mind, and to have a stable low-cost tree structure as objective.

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. The tradeoff is that HBH's tree construction algorithm is more complex than REUNITE's and thus HBH has a larger control overhead.

The results obtained through simulation shown that HBH outperforms REUNITE by 5% in terms of tree cost and 14% in terms of the delay experienced by the receivers, while having a control overhead up to 11%. We conclude that for unicast-based multicast protocols it is important to take into account the asymmetries of the underlying unicast routing infrastructure as these can impact the efficiency of the multicast trees constructed.

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] H. Holbrook and B. Cain, *Source-Specific Multicast for IP*, Mar. 2001. Work in progress, <draft-holbrook-ssm-02.txt>.

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

[6] L. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte, "Hop by hop multicast routing protocol," in *ACM SIGCOMM'2001*, pp. 249-259, Aug. 2001.

[7] B. Cain, S. Deering, W. Fenner, I. Kouvelas, and A. Thyagarajan, *Internet Group Management Protocol, Version 3*, Mar. 2001. Work in progress, <draft-ietf-idmr-igmp-v3-07.txt>.

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

[9] 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.

[10] 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.

[11] 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.

[12] V. Paxson, "End-to-end routing behavior in the internet," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 601-615, Oct. 1997.

[13] M. D. de Amorim, O. C. M. B. Duarte, and G. Pujolle, "Improving user satisfaction in adaptive multicast video", to appear in *Journal of Communications and Networks,* 2002.

[14] P. A. da S. Gonçalves, J. F. Rezende, O. C. M. B. Duarte, and G. Pujolle, "Optimal feedback for quality source-adaptive schemes in multicast multi-layered video environments", in *Networking'2002*, May 2002.

[15] 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.

[16] 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.

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