On Robustness in Application-Level Multicast: the case of HBM

Ayman EL-SAYED Vincent ROCA

INRIA Rhones-Alpes, Planète Project, France

{ayman.elsayed, vincent.roca}@inrialpes.fr

Abstract

This paper is devoted to an application-level multicast protocol, HBM (Host Based Multicast), which can be used when native multicast routing is not available. Being purely end-to-end, application-level multicast proposals in general, and HBM in particular, are intrinsically more fragile than traditional routing solutions relying on well administered and dedicated routers. Improving their robustness is therefore of high practical importance and we believe it is a key aspect for the acceptance of the technology by end users who won't tolerate that a multi-participant video-conference session be subject to frequent cuts. In this work we identify two classes of problems that lead to packet losses, and for each class we introduce and compare several schemes meant to improve the HBM robustness. Experiments show that in both cases simple yet efficient solutions to the problem exist.

1 Introduction

1.1 Application Level Multicast

Group communication traditionally requires that each node at each site has access to a native multicast routing service. If intra-domain multicast (within a LAN or a site) is widely available, this is different for inter-domain multicast (between several sites or ISPs). Today many ISPs are still reluctant to provide a wide-area multicast routing service [3]: there are technical reasons (many aspects are still research topics), marketing reasons (e.g. which pricing model) and an "egg and chicken" problem.

Application-level multicast proposals (see [5] for a survey of the main proposals) offer a practical solution to this problem. They enable every host to participate in multicast sessions, no matter whether it has access to native multicast or not. The HBM protocol [7, 4] is one such protocol. HBM is by nature centralized, everything being controlled by a single host, called Rendez-Vous Point (or RP).

1.2 The Robustness Issue and Related Works

Inter-domain multicast routing is often said to be fragile. If application level multicast offers a way to alleviate this problem, it also creates other instability problems. For instance, a solution based on end-hosts (usually PCs or workstations) is intrinsically less robust than one based on dedicated and well administered commercial routers. There is a high risk, as the group size increases, that the topology be partitioned after a single node failure.

Some proposals address robustness by using some level of flooding, like the gossiping approaches (Scribe [2]). If they offer a highly robust data distribution service, the difficulty is to estimate when to remove any given data item from the gossiping process. This scheme is therefore usually limited to small data transfers [5].

Many application level multicast proposals, comparable to HBM, merely content themselves with a fast detection and repair mechanism, for instance to identify partition problems and take counter measures [6]. In our opinion this reactive approach is definitively insufficient. For instance, some applications may require that partitions be avoided altogether (e.g. cooperative work or high quality multimedia-on-demand session) and reactive solutions are not acceptable.

An approach that shares similarities with our work is the Probabilistic Resilient Multicast (PRM) scheme [1]. Here a subset of the overlay tree nodes *randomly* "jump" data to other nodes of the tree, thereby creating redundant paths. Data coming from these random jumps is then flooded on sub-trees, unless it has already been received. A bit-mask indicating which packets have been received recently is piggybacked and offers the opportunity to ask for retransmissions to the node who jumped data. The random nature of the jumping process (which is the only feasible approach when no node has a consistent view of the topology) and the NACK/retransmission process are the main differences with our own solution.

The TMesh proposal [8] deliberately adds redundant links (called shortcuts) but with the goal of reducing latencies between members. A side effect is an increased robustness since shortcuts also provide redundant connections between members. But here also, since no single node has a consistent view of the topology, shortcuts are added in a random way (unlike HBM).

1.3 A Voluntary Approach to the Problem

In this paper we deliberately follow a voluntary approach: (1) by adding explicit redundancy in the overlay topology as well as a learning mechanism whereby less reliable hosts are identified and the topology created by taking it into account, and (2) by improving the topology update process which typically creates instability, and often losses. We regard robustness as a key aspect of HBM, and introduce proactive mechanisms that prevent, up to a certain point, packet losses.

The remaining of the paper is organized as follows: section 2 introduces the HBM proposal; section 3 explains how redundant virtual links can be added; section 4 explains how to reduce the probability of packet losses when updating the overlay topology; finally section 5 concludes this work.

2 Introduction to Host Based Multicast (HBM)

This section quickly introduces HBM (more details can be found in [7]) and the associated robustness problems.

2.1 Description

Basic Idea: The HBM protocol automatically creates a virtual overlay topology, which by default is a *shared tree*, between the various group members (sources and receivers), using point-to-point UDP tunnels. Everything is under the control of a single host, the *Rendez-vous Point* (RP). This RP knows the members, their features, and the communication costs between them. He is responsible of the distribution topology calculation and its dissemination among group members.

Periodic topology update: A dynamic adaptation of the overlay topology is required: (1) to reflect the changing networking conditions; (2) because of new members joining the group, who are initially grafted on the existing topology in a sub-optimal way; (3) after the departure of members (deliberately, after a crash, or because of a network failure); or (4) because recovery actions taken by the RP after a partition lead to a sub-optimal overlay topology. Therefore:

- all the members periodically evaluate the new communication costs between them (or a subset of them [4]) and inform the RP, and
- the RP periodically calculates a new topology and informs each member.

These two mechanisms are fully asynchronous.

Control messages: Several control messages are defined. In this paper we only consider the Topology Update (or TU) messages, sent by the RP to the members in order to inform them of the new topology. Since a TU message only contains the direct neighborhood, a different message is sent to each member.

2.2 The two Sources of Losses with HBM

Because the shared tree topologies created by default by HBM is an acyclic graph, if any transit member leaves the session[1], the tree gets partitioned. *Overlay topology partition is therefore the first source of packet losses.*

But *a second source of losses exists*. Since the overlay

[1] Only non-graceful leaves are considered here. During a graceful departure, the leaving node first contacts the RP who has the opportunity to take immediate measures.

topology must be periodically updated, and since it is impossible to guaranty the synchronism of the topology update process within each member[2], an instability period exists. During this period a subset of the members may be aware of (and use) the new topology, while others would only know (and use) the old one. Similarly, during this instability period, packets in transit may have been sent to either the new or old topology. Because of these transient routing problems, some packets may fail to reach all members which, from the application point of view, results in losses.

In the following sections we successively address both problems, introduce and compare several strategies.

3 Adding Redundant Virtual Links (RVL) to Avoid Topology Partition

3.1 Proposed Strategies

In order to reduce the probability of overlay topology partition in front of one or more node failures, we introduce Redundant Virtual Links (or RVL) [7] to the overlay topology created by HBM. Since the RP has a full knowledge of group members and creates/manages this topology, it can easily add a certain number of RVLs. The RVLs are *strategically placed* so as to provide some level of robustness guaranties (e.g. a resilience to any single node failure, or to any two simultaneous failures, etc.). In this work we only assume a "robustness to a single transit node failure" and evaluate experimentally the probabilistic robustness to several simultaneous node failures. This solution is not source dependent and therefore the robustness is the same no matter how many and where sources are.

More precisely we introduce and compare five different flavors, that differ on the way RVLs are added. For instance with the first flavor no difference is made between leaves and transit nodes, whereas the remaining four flavors limit the number of RVLs that can be attached to a leaf. This distinction makes sense since nodes with limited processing or communication capabilities are always moved toward the leaves of the overlay shared tree topology [7].

Strategy I: any number of RVLs can be attached to any node. No difference is made between leaves and transit nodes.

Strategy II: there is no limit on the number of RVLs attached to a transit node, but at most one RVL can be attached to a leaf. A RVL can be attached to any kind of node.

Strategy III: any number of RVLs can be attached to any node. A RVL cannot be attached to two leaves and only {leaf; transit node} and {transit node; transit node} RVLs are possible.

Strategy IV: there is no limit on the number of RVLs attached to a transit node, but no RVL can be attached

[2] It would require to freeze packet transmissions until the new topology is set up, an approach incompatible with real-time flows

to a leaf. Only {transit node; transit node} RVLs are possible.

Strategy V: there is no limit on the number of RVLs attached to a transit node, but at most one RVL can be attached to a leaf. Only {leaf; transit node} RVLs are possible.

The RVL addition algorithm follows a recursive approach. First find the two farthest nodes in the set, add a RVL between them, and split the set into two sub-groups, depending on their closeness to the two elected nodes. For each sub-group, do the same process, recursively, until the sub-group contains at most two nodes. The way the two farthest nodes are chosen depends on the strategy flavor mentioned above.

3.2 Examples

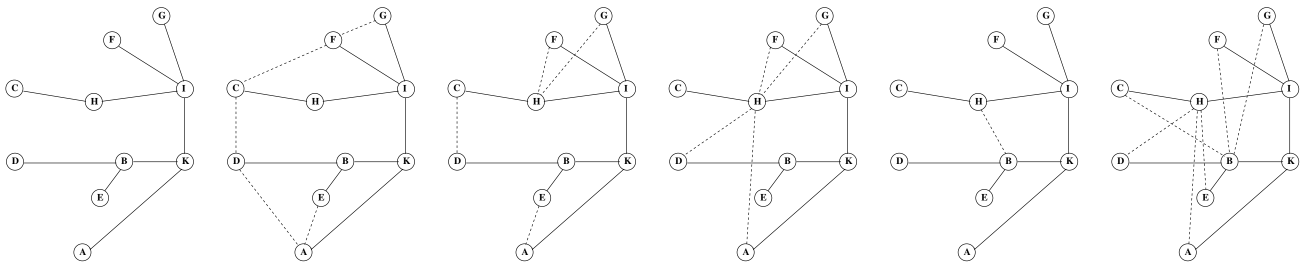
Let's consider a group of 10 members, and let's imagine the initial overlay topology created is given in Figure 1-a. The various strategies lead to the topologies of Figures 1-b/c/d/e/f. The interested reader can refer to the extended version for detailed explanations.

From this example we see that the number and the location of RVLs largely differ. For instance with strategy I we note that (1) the vast majority of RVLs are among leaf nodes, and (2) some leaf nodes have several RVLs attached. Therefore, strategy I is devoted to cases where all nodes have similar processing and communication capabilities, which is very restrictive. On the opposite strategy IV leads to the creation of a single RVL, and leaf nodes (who can be lightweight nodes, since the node features can be considered during the topology creation process) are never concerned by RVL.

3.3 Performance Evaluation Parameters

Let N be the total number of nodes. The following parameters are considered:

- the number of RVLs: N_{RVL}
- the ratio of the number of RVLs to the initial number of (non-RVL) links in the overlay: $R_{RVL} = \frac{N_{RVL}}{N-1}$. Note that with N nodes, without RVLs, there are always $N-1$ links in the shared tree overlay.
- the number of connected nodes after i node failures, respectively without and with RVLs: $R_{conn-without}(i)$ and $R_{conn-with}(i)$.
- the ratio of the number of connected nodes after i node failures to the total number of nodes, without and with RVLs: $R_{conn-without/with}(i) = \frac{N_{conn-without/with}(i)}{N}$. This ratio is an average over all possible sources (since each node can be a source in a shared tree). Ideally i node failures should leave $N-i$ connected nodes, so the maximum ratio is: $R_{conn-ideal}(i) = \frac{N-i}{N}$.



(a) Overlay (b) Strat. I (c) Strat. II (d) Strat. III (e) Strat. IV (f) Strat. V

Figure 1. Example of RVL addition (represented as dashed lines).

- the relative increase (or gain) in the number of connected nodes by adding RVLs, in front of i failures: $G_{conn}(i) = \frac{N_{conn-with}(i) - N_{conn-without}(i)}{N_{conn-without}(i)}$
- the average link stress: the stress is the number of identical copies of a packet carried by a physical link. This stress is evaluated with and without RVLs for all links and we consider the average value.

3.4 Experimental Evaluations

3.4.1 Experimental Conditions

The HBM protocol and all the previous strategies have been implemented. The experiments reported here are simulations based on a large interconnection transit-stub network, composed of 600 core routers, and generated by the Georgia Tech Model (GT-ITM) [9]. Some of these routers are interconnection routers, others, at the leaf of the topology, are access routers connecting the client sites. We then choose N sites randomly among the 243 possible leaves and compare each strategy.

3.4.2 Results and Discussions

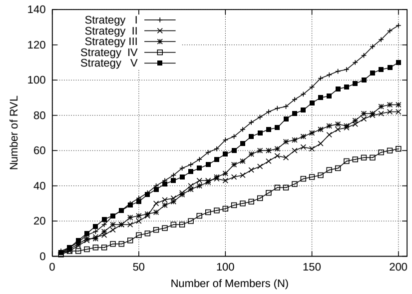
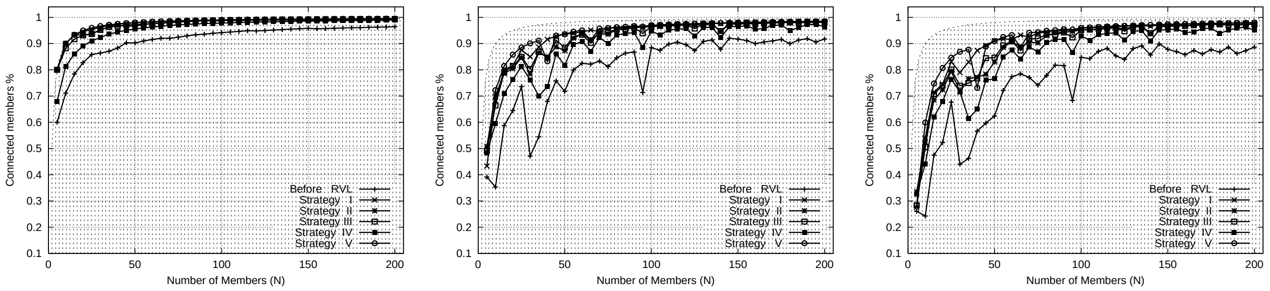


Figure 2. Number of RVLs added.

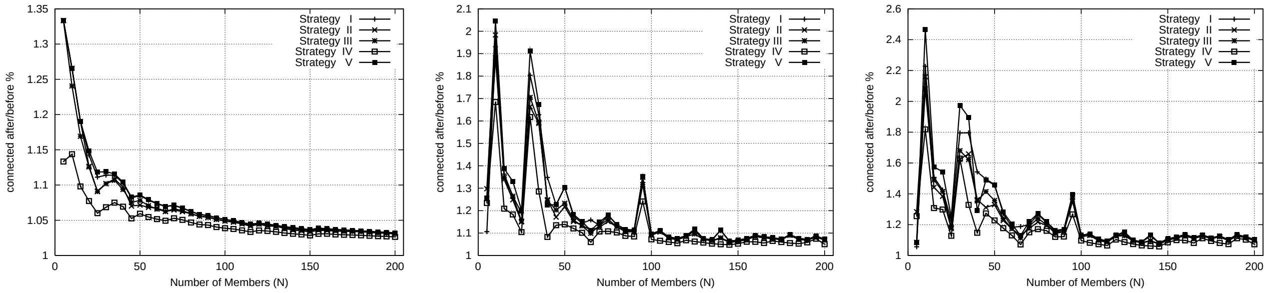
Figure 2 shows that the number of RVLs increases with N for all strategies, but with a different slope. Strategy IV adds the smallest number of RVLs, whereas strategy I adds the highest number of RVLs.

Figures 3-a/b/c depict the ratio of connected nodes when respectively one, two and three nodes fail. The upper edge of the dashed area represents the optimal ratio, $R_{conn-ideal}(i) = \frac{N-i}{N}$, where i is the number of failures. If all strategies that add RVLs improve the connectivity after a certain number of failures, we see that differences exist. Without any RVL, the ratio of connected nodes after a single failure amounts to 60% for small groups (5 nodes) and



(a) Single node failure (b) Two node failure (c) Three node failure

Figure 3. Ratio of connected nodes after 1, 2 or 3 failures, according to the RVL addition strategy.



(a) Single node failure (b) Two node failure (c) Three node failure

Figure 4. Relative increase in the number of connected nodes after 1, 2 or 3 failures, according to the RVL addition strategy.

96% for large groups (200 nodes). With strategy IV, the ratios are respectively 68% and 99%. With strategies I, II, III and V, the ratios are now 80% and 99%, which compares favourably to the 80% and 99.5% ideal values. We also note that for both two and three node failures, all strategies bring some benefits compared to the initial topology, even if the results are farther than the ideal.

Figures 4-a/b/c show the relative increase in the number of connected nodes, $G_{conn}(i)$. The benefits are significant and for groups having less than 100 members, are in a

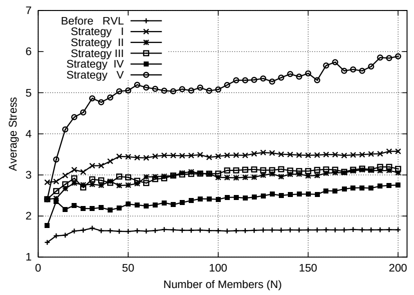


Figure 5. Average physical link stress.

$[4\%; 30\%]$ range. We note (1) that this gain decreases when N increases (larger groups), and (2) that this gain increases with the number of node failures. This second point is important and clearly argues for the systematic use of RVLs.

Finally figure 5 depicts the stress before and after adding RVLs. As expected, the stress increases when RVLs are added. Yet strategy IV has the smallest stress (because it has the smallest number of RVL) whereas strategy V adds a prohibitive amount of traffic.

To conclude we can say that *strategy IV offers a good balance between the robustness in front of non-graceful node failures, and additional traffic within the network. The gains are all the more significant as nodes are unstable. Besides the additional traffic is managed by transit nodes, never by leaves (who can have lower processing or networking capabilities).*

4 Reducing Packet Losses during Topology Updates

4.0.3 Parameters Affecting Losses

Several parameters affect the number of packets that can be lost during a topology update:

- the importance of changes: this is the number of links of the overlay that are modified and the number of nodes that are concerned by these modifications;
- the time required to inform all nodes concerned about the new topology: this parameter defines the period during which transient routing incoherences can occur;
- the number of packets in transit during this instability period: these are the packets that are potentially affected, i.e. that may be either lost (partition) or duplicated (loop).

Therefore routing problems will be all the more acute as the transmission rate is high, the changes numerous, and the topology update process long. It is the role of the RP to inform each node concerned by a topology update. If n nodes out of N are concerned, then the instability period lasts approximately:

$$T\_instability(n) \approx$$
$$\frac{n}{2} * duration\_of\_send + Max_{i \in 1..n}(delay\ to\ node\ i)$$

Indeed, even if the `send()` TCP socket function is called sequentially for all n nodes (hence the first part of the formula), transmissions take place in parallel, and the topology update message is available at the receiving application before the TCP segment has been acknowledged (hence the one way delay in the second part of the formula).

4.0.4 Proposed Strategies

Each strategy assumes that a topology be identified by a dedicated *globally unique and monotonically increasing Topology Sequence Number (TSN)*. This TSN is managed by the RP which guarantees its uniqueness. This TSN is present in all packets sent in the overlay topology, and each node remembers the current TSN in use along with its list of neighbors. Of course, because of the instability period, different members can have a different view of the current

TSN in use. Several strategies are then possible:

Strategy 1: Each time a node needs to update its topology he drops the current topology information (list of neighbors) and registers the new topology one. Each time a transit node receives a new packet having a TSN different from its own current TSN, either:

1(a): this packet is not forwarded. This default behavior is used as a reference in our experiments. It is a *conservative approach* that tries to limit the risk of creating loops at the cost of a higher packet loss rate.

1(b): the transit node forwards the packet over the current topology (except to the node from which it was received). This behavior tries to reduce the packet loss rate but increases the risk of creating loops.

1(c): if this packet has been received on a link that belongs to the current topology, this packet is forwarded, otherwise it is dropped.

Strategy 2: Each node keeps information for two topologies: the current one and the previous one. Each time a transit node receives a new packet, this latter is forwarded on the previous or current topology if its TSN is equal respectively to the previous or current TSN known by the node, and is dropped otherwise.

In all cases, each node keeps track of what packets have been received and drops duplicated packets in case a transient routing loop has been created.

4.1 Experimental Evaluations

4.1.1 Experimental Setup

Experiments take advantage of the environment mentioned in section 3.4.1. The importance of topology changes is controlled by the number of communication metrics that are changed (in practice we assign new random values to 25%, 50%, or 100% of the metrics). We do not simulate propagation delays between the various nodes and the RP but we check that the effective communication delays are in line with typical values (we measured from 1 ms to 1 second depending on the group size), and that the instability period is realistic (we measured from 100 ms to 540 ms). Tests are performed with 1024 byte packets and a 512 kbps transmission rate. It corresponds to 78.1 packets/s, which determines the number of packets in transit during the instability period. Each point in the figures is an average over 5 topology changes. Packet losses values are the average number of losses experienced by a node (i.e. the total number of packet losses divided by the number of nodes).

The quantitative results obtained are for sure highly dependent on the parameters chosen. We do not claim to have fully analyzed the problem space (in particular transmission delays between the various group nodes could be more realistically simulated using topology models). But we believe that the qualitative results obtained are realistic, which was our main goal.

4.1.2 Results and Discussions

Figures 6(a)-(c) show the average number of packet losses. We see that strategy 1(a) is the one which performs the worst in all cases, with around 4 packet losses. It does not depend on how much of the topology changed because even a single link change in the overlay trigger a TSN modification and all packets in transit with a wrong TSN are then dropped. Strategy 1(a), far too conservative, is definitely not appropriate.

Strategy 1(b) clearly improves robustness and for instance if the whole topology is changed, it leads to an average number of losses of approximately only 1 packet. The price to pay is a very high packet duplication ratio compared to other strategies (Figure 7), between 26% to 37%, whereas other strategies never exceed 4%.

Strategy 1(c) yields a better robustness than strategy 1(a) when the topology changes are small. Yet both strategies tend to be equivalent in case of important topology modifications.

Strategy 2 has excellent performances, both in terms of robustness and packet duplication. Moreover these performances depend neither on the group size nor on the importance of the topology update. Therefore we can conclude that *strategy 2 which keeps information for two topologies, is definitively the best solution and should be used systematically.*

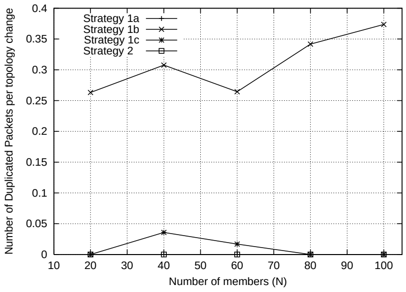


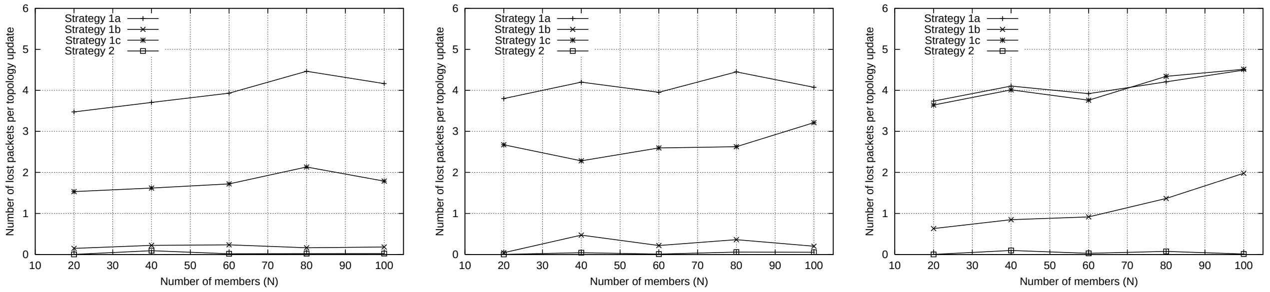
Figure 7. Number of duplicated packets per topology update (100% metrics changed).

5 Conclusions and Future Works

This paper focuses on various robustness aspects of an application level multicast proposal, HBM. We have identified two possible causes of packet losses: those caused by topology partition problems, usually after transit node failures, and those caused by routing instability periods, usually during the topology update process. We have introduced and compared several strategies, and experiments have shown that in both cases simple yet effective solutions exist.

Adding redundant virtual links to the overlay topology between a carefully chosen subset of transit nodes only is an easy way to improve robustness in front of node failures, even if a full robustness is not achieved. Going further requires to create RVLs between leaves and other nodes. This solution must take into account the exact capabilities of each node, since leaves are sometimes lightweight hosts with limited processing and networking capabilities.

A side effect of adding RVLs is an easy and rapid fail-



(a) 25% of metric change

(b) 50% of metric change

(c) 100% of metric change

Figure 6. Number of lost packets per topology update.

ure discovery capability: the fact that a node connected to a RVL receives new packets from the RVL only denotes a failure on the normal delivery path, and an alert message should be immediately sent to the RP in order to repair the partition. This solution is far more efficient than failure discovery mechanisms based on the periodic transmission of heartbeat messages between neighbors, since failure discovery is only possible at the end of each period, not immediately.

The second cause of packet losses is fully solved by the ”remember two topologies” strategy, which is clearly superior to the three other strategies. It should therefore be systematically used.

Future work will consider the possibility of reducing (or even suspending) the traffic sent on the RVLs, in order to reduce the stress generated. This can be easily achieved by sending on each RVL digests of recent data packets received, rather than a copy of these messages. This is reasonable when no failure takes place. In case of problem, a retransmission of these messages could be requested and copies of packets (rather than digests) could be once again sent on the RVL until the partition is recovered.

References

[1] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. In *ACM SIGMETRICS*, June 2003.

[2] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. Scribe: a large-scale and decentralised application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)(Special issue on Network Support for Multicast Communications)*, 20(8), October 2002.

[3] Christophe Diot, Brian Neil Levine, Bryan Lyles, Hassan Kassem, and Doug Balensiefen. Deployment issues for the ip multicast service and architecture. *IEEE Network*, pages 78–88, January 2000.

[4] Ayman El-Sayed and Vincent Roca. Improving the scalability of an application-level group communication protocol. In *10th Int. Conference on Telecommunications (ICT'03), Papeete, French Polynesia*, February 2003.

[5] Ayman El-Sayed, Vincent Roca, and Laurent Mathy. A survey of proposals for an alternative group communication service. *IEEE Network, special issue on Multicasting: an enableing technology*, January/February 2003.

[6] Yang hua Chawathe, Sanjoy G. Rao, and Hui Zhang. A case for end system multicast. In *ACM SIGMETRICS*, June 2000.

[7] Vincent Roca and Ayman El-Sayed. A host-based multicast (hbm) solution for group communications. In *First IEEE International Conference on Networking (ICN'01), Colmar, France*, pages 610–619, July 2001.

[8] Wenjie Wang, David Helder, Sugih Jamin, and Lixia Zhang. Overlay optimizations for end-host multicast. In *In proceeding of the fourth International Workshop on Networked Group Communication (NGC 2002), Boston, USA*, October 2002.

[9] E. Zegura, K. Calvert, and S. Bhattacharjee. How to model an internetwork. In *Proceedings of IEEE INFOCOM'96*, March 1996.