# A Comparison of Application-Level and Router-Assisted Hierarchical Schemes for Reliable Multicast

Pavlin Radoslavov[†], Christos Papadopoulos[†], Ramesh Govindan[†], Deborah Estrin[‡]

*Abstract*—**One possible approach to achieve scalability in reliable multicast is to use a hierarchy. A hierarchy can be established at the application level, or by using router assistance. Because the routers have more detailed knowledge about the underlying network topology, intuitively a hierarchy produced by router assistance is expected to improve performance. In this paper we test this assumption by comparing two schemes, one that uses an Application-Level Hierarchy (ALH) and another that uses Router-Assisted Hierarchy (RAH). Contrary to our expectations, we find that the qualitative performance of ALH is comparable to RAH when a good technique is used to build the hierarchy. We do not model the overhead of creating the hierarchy nor the cost of adding router-assistance to the network. Therefore, our conclusions inform rather than close the debate of which approach is better.**

*Keywords*—**Reliable Multicast, Router Assistance for Reliable Multicast**

## I. Introduction

Reliable multicast has received significant attention recently in the research literature [3], [14], [11], [6], [2], [10], [8], [17]. The key design challenge for reliable multicast is scalable recovery of losses. The two main impediments to scale are *implosion* and *exposure*. Implosion occurs when, in the absence of coordination, the loss of a packet triggers simultaneous redundant messages (requests and/or retransmissions) from many receivers. In large multicast groups, these messages may swamp the sender, the network, or even other receivers. Exposure wastes resources by delivering a retransmitted message to receivers which have not experienced loss. Another challenge that arises in the design of reliable multicast is long *recovery latency*, which may result from suppression mechanisms to solve the implosion problem. Latency can have significant effect on application utility and on the amount of buffering required for retransmissions. Finally, highly dynamic groups may result in a loss of efficiency because they break assumptions about group constituency and structure.

One popular class of solutions is *hierarchical data recovery*. In these schemes, participants are organized into a hierarchy. By limiting the scope of recovery data and control messages between parents and children in the hierarchy, both implosion and exposure can be substantially reduced. Hierarchies introduce a latency penalty, but that only grows proportional to the depth of the hierarchy. The biggest challenge with hierarchical solutions is the construction and maintenance of the hierarchy, especially for dynamic groups. For optimal efficiency, the recovery hierarchy must be *congruent* with the actual underlying multicast tree [1]. Divergence of these structures can lead to inefficiencies when children select parents who are located downstream in the multicast tree.

One approach, exemplified by RMTP [11], is to use manual configuration or application-level mechanisms to construct and maintain the hierarchy. Manual hierarchy construction techniques rely either on complete or partial (*e.g.*, where the border routers are) knowledge of the topology. Automated hierarchy construction techniques rely on dynamically discovering tree structure, either explicitly by tracing tree paths [10], or implicitly by using techniques based on expanding ring search. Once a hierarchy is formed, children recursively recover losses from their parents in the hierarchy by sending explicit negative acknowledgments.

Another approach, exemplified by LMS [14], proposes to use minimal router support not only to make informed parent/child allocation, but also to adapt the hierarchy under dynamic conditions. In some of these router-assisted schemes, hierarchy construction is achieved by routers keeping minimal information about parents for downstream receivers, then carefully forwarding loss recovery control and data messages to minimize implosion and exposure. In these schemes, hierarchy construction requires little explicit mechanism at the application-level at the expense of adding router functionality. Because of this, one would expect these *router-assisted hierarchies* (Section II-B) to differ from the *application-level hierarchies* (Section II-A) in two different ways: a) router-assisted hierarchies are *finer-grained*; that is, have many more "internal nodes" in the hierarchy; and b) they are more congruent to the underlying multicast tree.

Then, it is natural to ask, as we do in this paper: Is the performance of application-level hierarchies qualitatively different than that of router-assisted hierarchies? To our knowledge, this question has not been addressed before. We study this question by evaluating two specific schemes: LMS and an RMTP-like schemes which use two specific hierarchy construction techniques. For our comparison we used four metrics: recovery latency, exposure, data traffic network overhead, and control traffic network overhead. We approach the question from two angles: first, we use analysis (Section III) to determine the asymptotic behavior of the two schemes for regular trees, and second, we employ simulation (Section IV) to study the performance of large irregular multicast trees. These irregular multicast trees

[†] P. Radoslavov, C. Papadopoulos and R. Govindan are with the USC/Information Sciences Institute, 4676 Admiralty Way, Suite 1001, Marina del Rey, CA 90292-6695, USA. E-mail: pavlin@catarina.usc.edu, {christos,govindan}@isi.edu.

[‡] D. Estrin is with UCLA and Information Sciences Institute. E-mail: estrin@isi.edu

This work was supported by the National Science Foundation under Cooperative Agreement NCR-9321043. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

[1]Congruency is achieved when the virtual hierarchy and the underlying multicast tree coincide.

are randomly generated on real-world topologies such as the Internet [4], and the Mbone [13] topology [16].

Before doing this performance comparison, our expectation was that router-assisted hierarchies would significantly outperform application-level hierarchies. Our finding was surprising: that, with careful hierarchy construction, the performance of application-level hierarchies *is comparable* to that of router-assisted hierarchies, even though the former have a coarse-grained recovery structure. However, as we show, there exist pathological hierarchy construction techniques for which application-level hierarchies perform qualitatively worse than router-assisted hierarchies. Thus, the congruence of the hierarchy to the underlying multicast tree seems to be more important to performance that having a fine-grain recovery structure.

We should emphasize that we model only the essential features of the two schemes, and while our conclusions may be colored by the specific schemes we chose, we believe our results have a bearing on the larger issue of how router-assisted hierarchies compare to application-level hierarchies. Furthermore, our conclusions inform but do not necessarily close the debate regarding the appropriate approach to hierarchical data recovery. Our evaluation metrics do not capture the complexity and cost of hierarchy construction, or the complexity of adding router-assistance for hierarchical recovery to the network.

The rest of the paper is organized as follows. In Section II we present in details the application-level and router-assisted schemes we consider in this paper, and describe the evaluation metrics. Section III presents the k-ary tree analytical results for both schemes. Section IV present and discusses the simulation results for real-network topologies. Related work and conclusions are in Section V and Section VI respectively.

## II. Hierarchical Multicast Data Recovery Schemes

As the name implies, hierarchical reliable multicast schemes solve the scalability problem by structuring the group into a hierarchy. Because a hierarchy explicitly enforces scope on the data recovery, it is a natural approach to address many of the problems described earlier, including implosion, exposure and latency. Based on the mechanisms used to create and maintain the hierarchy, we can distinguish between two classes of hierarchical schemes. The first class, *application-level hierarchical schemes (ALH)*, uses only end-to-end mechanisms assisted by the end-systems (the receivers) to create and maintain the hierarchy. The second class, *router-assisted hierarchical schemes (RAH)*, uses assistance from the routers in the creation and maintenance of the hierarchy.

### A. Application-Level Hierarchical Schemes

ALH schemes create and maintain the data recovery hierarchy by using only end-to-end mechanisms. Typical mechanisms include manual (static) configuration and expanding ring search to locate the nearest candidates. More sophisticated schemes employ heuristics like “loss fingerprinting” where receivers compare their loss fingerprints with those of potential parents and select the most appropriate. Both types, however, tend to be slow to adapt to dynamic conditions and are not always accurate in maintaining congruency.

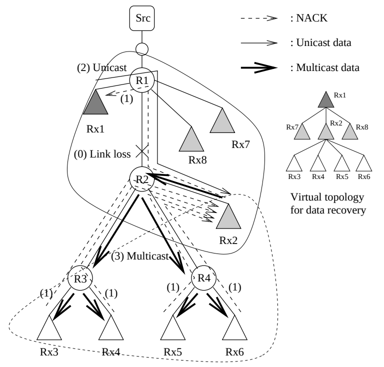

Fig. 1. ALH example: optimal hierarchy organization

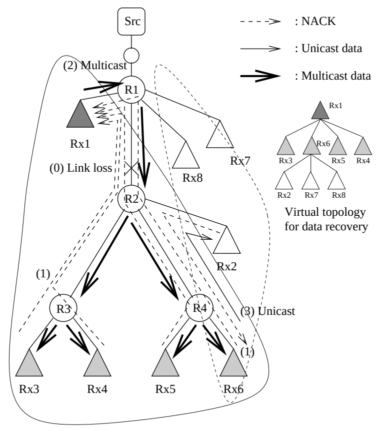

Fig. 2. ALH example: sub-optimal hierarchy organization

The Reliable Multicast Transport Protocol (RMTP) [11] is an example of an ALH scheme, and forms the basis of our ALH model. RMTP employs a combination of positive and negative acknowledgments (ACK and NACK)for data recovery. Because the focus of this paper is not on modeling and evaluating protocol details, but rather understanding the underlying mechanisms, we do not follow exactly the RMTP protocol; instead, we adopt the hierarchical approach in RMTP and model only NACKs and retransmissions.

Briefly, our ALH scheme works as follows. On Figure 1 $Rx1$ is a parent for $Rx2$, $Rx7$, and $Rx8$, while $Rx2$ itself is a parent for $Rx3$, $Rx4$, $Rx5$ and $Rx6$. Upon detecting loss (link $R1 - R2$), children unicast NACKs to their parents ($Rx3$,$Rx4$, $Rx5$, $Rx6$ to $Rx2$, and $Rx2$ to $Rx1$.) If the parent has the data it sends it to its children by either unicast or multicast. A multicast response is sent to a local multicast group where only the children and the parent are members of this group. To select between unicast and multicast, a parent collects NACKs and uses multicast if at least 50% of the children requested data retransmission; otherwise the parent uses unicast (parent $Rx1$ to $Rx2$.) If a parent does not have the requested data, its own parent also detects the loss from missing acknowledgments (and so on until we reach the root). After receiving the data each parent sends it to its children.

RMTP does not explicitly specify how the hierarchy is created; rather, in its current incarnation it assumes a manually configured static hierarchy. In order to explore the potential of ALH schemes, we introduce a rather powerful heuristic: we hypothesize that all participants have somehow obtained information about the distance to each other, and use that information in a heuristic algorithm to create the hierarchy. The algorithm creates the hierarchy in a bottom-up fashion as follows: among a group of participants, the node with the smallest sum of distances to all other nodes becomes a parent. Initially, all receivers are eligible to become parents and thus the lowest level of the hierarchy is formed by selecting parents among all receivers. Each of the receivers which was not elected as a parent chooses the closest parent node as its parent. The same heuristic is recursively applied at the next level among all the nodes that were selected as parents in the previous iteration, until we are left with a small number of nodes which become children of the root of the tree (*i.e.*, the sender). The depth of the hierarchy is defined by the fraction of nodes to choose at each iteration, which is a number in the interval $(0.0, 0.5)$. A value of 0.1 for example means that among all nodes at level $i$ in the data recovery hierarchy, 10% of them will become parents of the remaining 90%.

## *B. Router-Assisted Hierarchical Schemes*

Router-assisted hierarchical schemes (RAH) use network assistance to achieve congruency between the hierarchy and the multicast tree. By eliminating the need to maintain the hierarchy through potentially expensive and complicated endsystem-based mechanisms, RAH schemes reduce application complexity and enable the development of large-scale reliable multicast applications. For our evaluation of RAH schemes we chose Lightweight Multicast Services (LMS) [14] as our model. LMS employs router assistance to create a dynamic hierarchy which continuously tracks the underlying multicast routing tree regardless of membership changes. The network assistance required by LMS is in the form of new forwarding services at the routers, and thus has no impact at the transport level. With LMS each router marks a downstream link as belonging to a path leading to a *replier*. A replier is simply a group member willing to assist with error recovery by acting as a parent for that router's immediate downstream nodes. Because they are selected by routers, parents are always upstream and close to their children. The forwarding services introduced by LMS allow routers to steer control messages to their replier, and allow repliers to request limited scope multicast from routers. More specifically, LMS adds the following three new services to routers:

• *Replier selection:* potential repliers advertise their willingness to serve as repliers for a particular *(Source, Group)* pair with their local router. Routers propagate these advertisements upstream. Before propagating the message upstream, a router selects one of its downstream interfaces (based on an application-defined metric) as the replier interface. When all routers have received advertisements the replier state is established. Replier state is soft state which provides robustness and guards against replier and link failures.

• *NACK forwarding:* LMS routers forward NACKs hop-by-hop according to the following rules: a NACK from the replier interface is forwarded upstream; a NACK from a non-replier interface (including the upstream interface) is forwarded to the replier interface. However, a NACK from a non-replier downstream interface marks this router as the "turning point" of that NACK. Note that by definition, there can be only one turning point for each NACK but the same turning point may be shared by multiple NACKs. Before forwarding a NACK, the turning point router inserts in the packet the addresses of the incoming and outgoing interfaces, which we call the "turning point information" of the NACK. This information is carried by the NACK to the replier.

• *Directed multicast (DMCAST):* DMCAST is used by repliers to perform fine-grain multicast. A replier creates a multicast packet containing the requested data and addresses it to the group. The multicast packet is encapsulated into a unicast packet and sent to the turning point router (whose address was part of the turning point information) along with the address of the interface the NACK originally arrived at the turning point router. When the turning point router receives the packet, it decapsulates and multicasts it on the specified interface. An enhanced version of DMCAST may allow repliers to specify more than one interface that the packet should be directed to send on.

LMS works well in most cases to deliver the requested packet with minimum latency and only to receivers that need it. Figure 3 shows such an example. The loss on link $R1 - R2$ is recovered from replier $R1$ by sending a DMCAST to $R1$. How-

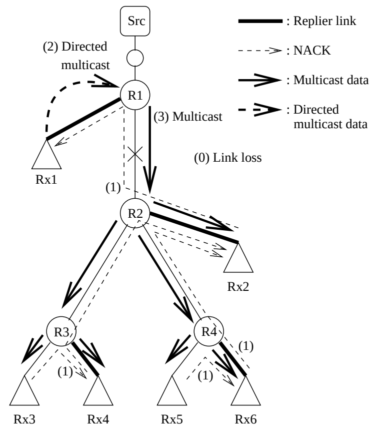

Fig. 3. LMS vanilla example: data loss and recovery

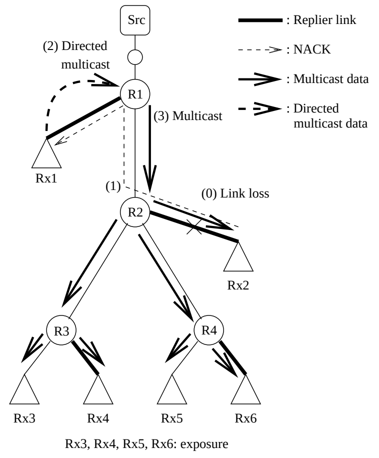

Fig. 4. LMS vanilla example: data loss by replier only and exposure to other receivers

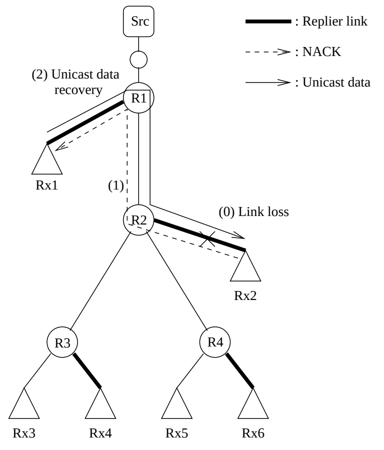

Fig. 5. LMS enhanced example: data loss by replier only and unicast recovery

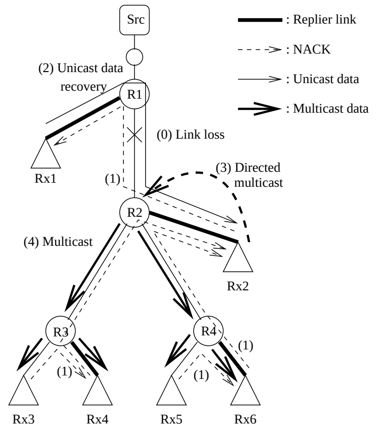

Fig. 6. LMS enhanced example: two-step data recovery (unicast followed by direct multicast)

ever, in some cases LMS may expose receivers to retransmissions that do not need it. This occurs when loss happens on the replier path, as shown in Figure 4. The resulting exposure does not affect correctness but may lead to wasted resources if a replier branch (the link between $R2$ and $Rx2$ in our example) is particularly lossy. LMS addresses this problem by selecting the replier branch that advertises the least loss. However, determining path loss characteristics can be hard, and thus LMS employs another method to eliminate exposure, which comes at the cost of eventually adding an extra hop to the retransmission. With this enhancement, a NACK by a downstream replier specifies that the reply should be unicast to the requestor itself rather than the turning point. For example, in Figure 5 (the same loss scenario as in Figure 4) $Rx1$ will directly unicast the reply to $Rx2$ and therefore there will be no exposure on the subtrees rooted at $R3$ and $R4$. The extra hop of retransmission can be illustrated by the example on Figure 6 and where the packet loss occurs on the link between $R1$ and $R2$. Similar to the previous example, the request by downstream replier $Rx2$ will reach $Rx1$ and the reply will be unicast back to $Rx2$. However, because in the mean time $Rx2$ has received NACKs from the downstream parts of the tree ($Rx4$ and $Rx6$), now it just needs to send a single enhanced directed multicast to $R2$ specifying that the reply should be multicast on links $R2 - R3$ and $R2 - R4$. Only if at some later time the requestor $Rx2$ receives more requests, direct multicasts are sent to the remaining part of the subtree. Note that this two-step process occurs only once, between the replier above the loss and the first requestor. We distinguish the previous version (which we call vanilla LMS) from this version, which we call “enhanced LMS.” Preliminary experiments have shown that for large groups the increase in latency in enhanced LMS is negligible but there is a significant reduction in exposure. Therefore, in this paper we use the enhanced version of LMS.

We note that LMS is not the most aggressive router-assisted recovery scheme. Finer grain recovery schemes, in which routers themselves respond to loss recovery requests from downstream neighbors, can, perhaps perform better than LMS. While such schemes are conceivable, we believe they are impractical in that they require significant router state.

## C. *Metric Space*

We did not model the overhead of creating a hierarchy with ALH schemes, because this depends strongly on the application and network characteristics. For example, in an application where membership is static, parents can be deployed manually and can yield excellent performance. At the other extreme, applications with mobile receivers may impose many restrictions on the type of the hierarchy creation algorithm and the parent–child associations.

If we ignore the overhead for creating and maintaining the hierarchy, the main source of inefficiencies in ALH schemes is the lack of congruency between the possibly fine grain hierarchy and the underlying multicast tree. Divergence of the two structures results in problems when children inadvertently join parents that are located downstream or are too far away, which results in an increase in recovery latency and network overhead. For example, on Figure 2 $Rx6$ is at the very bottom of the multicast delivery tree, but is inadequately a parent for the upstream $Rx2$, $Rx7$ and $Rx8$. RAH schemes do not suffer from these problems because they continuously track the multicast topology (although the cost varies among different schemes).

We have defined a set of metrics to evaluate the two schemes which represent the impact of the data recovery mechanism on the application and the network. These metrics are defined below; in section II-D we will present some examples of those metrics computed for different data recovery schemes.

• **Data recovery latency**. The recovery latency is defined as the ratio of the data recovery time observed by a receiver and the round-trip time from that receiver to the sender. A smaller value means that the receivers will wait shorter time to receive the missing data. For example, latency of 0.5 means that the time it will take for the receiver to recover the data is half of the round-trip time to the root/sender. The formula we use to compute the average data recovery latency across all receivers and across all links being lossy is:

$$NormLat = \frac{\sum_{receivers(r)} \sum_{links(l)} \frac{Lat(r,l)}{RTT(r)}}{NumberOfLossRcvs * NumberOfLinks}$$

where $Lat(r, l)$ is the receiver $r$ latency when the packet loss is on link $l$, $RTT(r)$ is the round-trip time from receiver $r$ to the root of the tree, $NumberOfLossRcvs$ is the total number of receivers that have observed any loss, and $NumberOfLinks$ is the total number of links in the topology.

In ALH schemes, sources of latency include longer recovery paths due to lack of congruency between the hierarchy and the multicast tree, and the latency due to multiple hops (parents). RAH schemes typically do not suffer from these problems because they (a) almost always recover from the nearest replier, and (b) have the capability of sending the multicast data to only one branch of the tree.

• **Receiver exposure**. The exposure is defined as the ratio of the extra amount of packets that have been received by a receiver (and eventually discarded), and the total number of packets sent by the sender. Ideally, this metric should be 0 (*i.e.,* no extra packets received and no extra processing by the receivers). The formula we use to compute the average exposure across all receivers and across all links being lossy is:

$$NormExp = \frac{\sum_{receivers(r)} \sum_{links(l)} Exposure(r,l)}{NumberOfExpRcvs * NumberOfLinks}$$

where $Exposure(r,l)$ is the exposure for receiver $r$ (in term of number of extra packets) when the packet loss is on link $l$, $NumberOfExpRcvs$ is the total number of receivers that have observed any exposure, and $NumberOfLinks$ is the total number of links in the topology.

ALH schemes suffer from exposure when more than half (but not all) children lose a packet which result into a local multicast. With RAH schemes the problem is limited to only few specific cases due to the ability to use subcast.

• **Data traffic network overhead**. The data traffic network overhead is defined as the ratio of the amount of used network resources because of the retransmitted multicast data (in term of total number of data packets sent over any link in the network), and the size of the subtree (in number of links) that did not receive the data. In the ideal case the data network overhead will be 1.0 (*e.g.,* when the node right above the lossy link has the data and it will send a single multicast packet down the whole branch of the tree that did observe the loss). ALH schemes suffer from this overhead because of the inefficiency introduced by the unicast/multicast combination. The formula we use to compute the average data overhead across all lossy links is:

$$NormDataOverhead = \frac{\sum_{links(l)} \frac{Data(l)}{Subtree(l)}}{NumberOfLinks}$$

where $Data(l)$ is the total amount of data traffic that will be created when the packet loss is on link $l$, $Subtree(l)$ is the size of the subtree (in term of number of links) that did not receive the data, *i.e.,* the subtree below (and including) link $l$, and $NumberOfLinks$ is the total number of links in the topology.

• **Control traffic network overhead**. Similar to the data traffic overhead, the control overhead is defined as the ratio of the amount of used network resources by the control packets (the NACKs) and the size of the subtree that did not receive the data. We consider ratio of 1.0 as optimal, even though this is not the theoretically lowest ratio. For example, if the node right above the lossy link was a replier, the control overhead will be 1.0 if there was exactly one NACK sent over all of the links of the subtree below the lossy link before the replier receive a NACK. ALH schemes may suffer more than RAH schemes because with ALH there is less opportunity to do NACK fusion. Similar to the data overhead, the formula we use to compute the average control overhead is:

$$NormControlOverhead = \frac{\sum_{links(l)} \frac{Control(l)}{Subtree(l)}}{NumberOfLinks}$$

where $Control(l)$ is the total amount of control traffic that will be generated in the network when the packet loss is on link $l$.

### *D. Examples of Measuring ALH and RAH Performance*

The metrics we described in the previous section can be illustrated by the following examples. Consider first the ALH example on Figure 1. Five of the receivers will send NACKs to their parents, and the control overhead will be $\frac{15}{8} = 1.875$ (the size of the subtree that did not receive the data is 8). The data overhead then will be $\frac{3+7}{8} = 1.25$. The data latency for receiver $Rx2$ will be 6 (the RTT to $Rx1$), but the latency for $Rx3$, $Rx4$, $Rx5$, $Rx6$ will be $-1+6+3 = 8$ [2]. If we assume that the sender is two hops away from $R1$, then the average data latency will be $\frac{6/8+4*8/10}{5} = 0.79$. The exposure in this particular example is 0. If the ALH data recovery hierarchy was not created efficiently, such as the hierarchy on Figure 2, then the latency, data and control overhead will be respectively 0.94, 1.375, and 2.375 (note that the latency for $Rx2$ is 12, because it is one hop closer to the sender than its parent). The exposure in this example is also 0.

If we look in the example for the RAH scheme on Figure 6 (enhanced LMS) which has the same setup of receivers and link loss as in the above example, the latency, data and control overhead, and exposure are respectively 0.79, 1.25, 1.625 and 0. On the contrary, if we used vanilla LMS (see Figure 3), the latency, data, and control overhead would be respectively 0.63, 1.125 and 1.625. However, the average receiver exposure on Figure 4 for each of the receivers that received extra packet ($Rx3$, $Rx4$, $Rx5$, $Rx6$) will be $\frac{4*(2-1)}{4} = 1.0$.

## III. K-ARY TREE ANALYSES OF RAH AND ALH

To get initial understanding on the scalability property of the ALH and RAH schemes, we did some simple analyses on k-ary trees. In our analyses we assumed that the root of the tree is the sender, and that all leaves are receivers. Thus, a k-ary tree of depth $L$ has $k^L$ receivers. Also, in case of ALH we assumed that the recovery hierarchy is created such that each parent has $k-1$ children. Given these assumptions, the same parent nodes for ALH are the repliers for RAH. For each of the schemes we assumed a single link loss and computed the average (per link-loss across all links) for each of the metrics described in Section II-C.

Table III shows the approximate analytical results [3]. The results for the latency, data and control overhead are plotted on Figure 7, Figure 8 and Figure 9 respectively. We should note

[2] Receiver $Rx2$ will discover the data loss and will initiate the recovery one "link-hop" time unit earlier than $Rx3$, $Rx4$, $Rx5$ and $Rx6$, hence the $-1$ in the latency computation.

[3] The assumption for the approximation is that $L^2 << k^L$.

| Metric | RAH | ALH |
|---|---|---|
| Latency | $\approx \frac{3k-1}{L(k-1)} + \frac{3}{2k} - \frac{9}{kL}$ | $\approx \frac{3}{2} + \frac{L}{6} - \frac{L+3}{6k}$ |
| Exposure | 0 | 0 |
| Data Overhead | $\approx 2 + \frac{1}{k} + \frac{1}{k^2}$ | $\approx 2 + \frac{1}{k} + \frac{5}{k(k+1)}$ |
| Control Overhead | $\approx 2 + \frac{4k-2}{k^2(k+1)}$ | $\approx 2 + \frac{4k-2}{k^2(k+1)}$ |

TABLE I

K-ARY TREES ANALYTICAL RESULTS (AVERAGE).

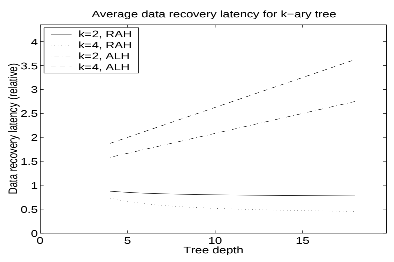

Fig. 7. RAH and ALH: average data recovery latency

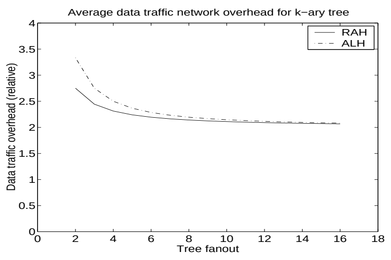

Fig. 8. RAH and ALH: average network data overhead

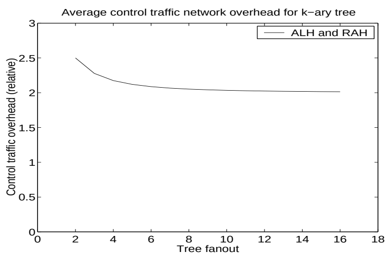

Fig. 9. RAH and ALH: average network control overhead

that it is not because of the approximation, but by the setup definition that the control overhead for both schemes is the same, and the exposure is zero.

First, it is interesting to note that the data and the control overhead for both schemes are independent (after the approximation) from tree depth $L$, and only slightly depend on tree fanout $k$, approaching factor of two overhead for large fanout. The factor of two is because, for example, in most cases a loss on a receiver link will be recovered by a parent/replier that is just two hops away. Further, the data overhead for ALH is slightly worse than RAH. This is a little bit surprising, because intuitively one would have expected a larger difference: in RAH a single subcast will cover a whole branch of the tree, while in ALH the recovery will be multicasted to only some of the nodes within that branch (the children of the parent performing the recovery, and then re-multicasted by those children.) On the other hand, while the data recovery latency for RAH decreases slightly when the tree depth $L$ increases, the ALH latency increases linearly with $L$. However, the increase is in fact logarithmic to the number of receivers, and only for extremely large number of receivers the difference between RAH and ALH can be of the order of few times.

## IV. SIMULATION RESULTS

The analytical results we presented in Section III apply only given the assumptions we have about the topology and the receivers setup, and may not be true in the general case. Some of the questions we want to answer through numerical simulations are:

- How RAH and ALH perform with real-world router-level topology and how do they compare to each other.
- How other real-world topology may impact the results.
- What is the impact of the hierarchy creation parameter for the ALH scheme.
- What would be the performance penalty for ALH if we did not use any heuristic to create the data recovery topology (*i.e.*, if the hierarchy was randomly created).

First we will describe our simulation setup, and then will present and discuss the results.

### A. Simulation Setup

In most of the simulations we used a router-level Internet-core topology of 54533 nodes [4]. To investigate the results sensitivity for different real-world topologies, we used also a map of the Mbone [13] topology (a virtual router-level overlay topology) with total of 4179 nodes [16]. We assumed a single-source multicast distribution tree with the source at the root of the tree. The receivers were placed at random and their number varied between 5 and 5000 (2000 for Mbone). The default hierarchy creation parameter for the ALH scheme was 0.1, *i.e.*, on average each parent had 9 children (1/0.1 - 1). Further, to prevent extremely uneven distribution of the children among the parents, the maximum number of children a parent may have at each level was set to $(4 * (1/frac_{pc} - 1))$, where $frac_{pc}$ is the hierarchy creation parameter. Some of the results we present for ALH are both with the inter-receiver distance heuristic we described in Section II-A, and without any heuristic (named ALH-heuristic and ALH-random respectively). In the second case, the set of parents selected at each level of the hierarchy is completely random, and then each child chooses randomly the parent to connect to. The results for ALH-random would eventually give us the worst-case ALH performance, *i.e.*, when we do not have a good mechanism to create the recovery hierarchy. For each set of parameters we performed 50 simulations with a different set of receivers [4]. The results we present are averaged across all simulations, but we also present the 95% confidence interval (even though in most cases this interval is very small to be noticed).

For each scheme we measured the data recovery latency, exposure, data overhead, and control overhead across all links go-

[4]We did some experiments with a larger number of receiver sets but in all simulations there was relatively small variation in the results.

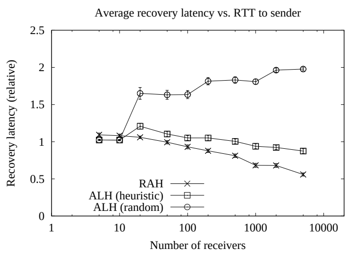

Fig. 10. RAH and ALH: average data recovery latency

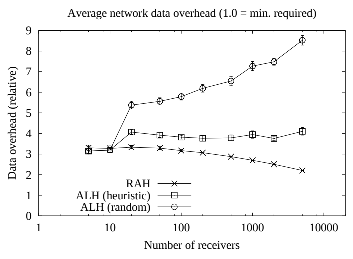

Fig. 12. RAH and ALH: average network data overhead

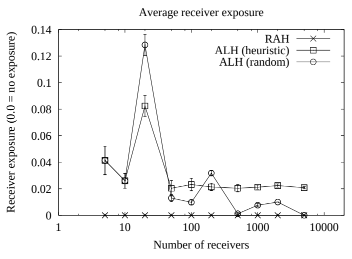

Fig. 11. RAH and ALH: average receiver exposure

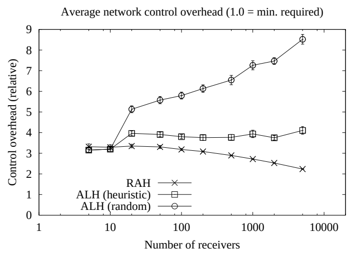

Fig. 13. RAH and ALH: average network control overhead

ing down (one-by-one). For simplicity we assumed that all links have the same propagation latency, and that sending a single packet over any of the links creates the same overhead to the network. The measured results were averaged across all links [5]. The metrics were computed using the expressions in Section II-C.

As we mentioned in Section I, we are not interested in investigating the particular protocols in details, but only in the underlaying schemes instead. For this reason we did not include in the basic schemes various protocol enhancements such as multiple LMS router state for routers with large fanout [14] that can help to reduce the control overhead.

### B. RAH and ALH Simulation Results

Figure 10 shows the data recovery latency for RAH and ALH (with and without the hierarchy creation heuristic). First, we can see that the results for RAH did match our analytical results. The fact that the RAH latency decreases when the number of receivers increase can be explained by the simple observation that a larger number of receivers increases the probability that there is a closer replier that has received the data, and therefore the recovery latency will be shorter. Surprisingly, the ALH-heuristic results were very similar to the RAH results but did not match our analytical results. This can be explained by the fact that in the k-ary trees there is strict enforcement on the recovery hierarchy construction (*i.e.*, a parent can only be a leaf node), while in real-world topologies our heuristic will quite likely choose for each child/receiver its parent node to be reasonably close on the shortest path to the root. It is quite likely that such node will be chosen as a replier in RAH, and therefore the results for both schemes are similar. On the other hand, it is less likely that in ALH-random the parent will be on the shortest path. Hence, when the number of receivers increase, the number of levels in the data recovery hierarchy which do not follow the shortest path between the sender and each receiver will increase as well, and therefore the receiver latency will be longer.

[5]We also experimented with weighting the results by the link loss probability which we assumed is proportional to the number of end-to-end shortest paths that use each link, but the results were similar.

Figure 11 presents the results for the receiver exposure. The RAH exposure is always zero by the mechanism definition (true for a single link loss, but may not always be true if there were multiple link losses). The results for both ALH-heuristic and ALH-random are reasonably low. Surprisingly, ALH-heuristic performed worse than the ALH-random. The reason for this is that in ALH we can have exposure only if the parent uses multicast to send the data to its children. In our simulations the parent would use multicast only if at least 50% of the children did not receive the data. With ALH-heuristic there is larger probability for children locality, and therefore if any of them did not receive the data, there is larger probability that at least 50% of its siblings did not receive it either (*i.e.*, larger probability that the parent will use multicast).

Figure 12 and Figure 13 show the results for data and control overhead respectively. Here again the results for RAH and ALH-heuristic are very similar. However, while the RAH results match the analytical results, it is difficult to say the same thing for the ALH. Similar to the latency, the ALH-random results show that the overhead increases for a larger number of receivers, an artifact from the increased average depth of the recovery tree.

We should note that for all simulation the data and the control overhead seemed to be almost identical. On closer examination, the RAH control overhead was approximately 5-10% higher than the data overhead. We can explain the reason for this small difference by the fact that there is extra control traffic only over the path between a router-turning point and its replier, a path that by definition is as short as possible for that router, therefore the control overhead is minimized. Indeed, this overhead can be up to $O(RouterFanout)$, but in most cases it did

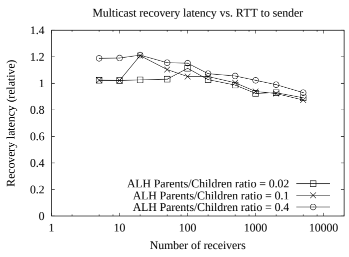

Fig. 14. ALH: latency sensitivity to hierarchy organization

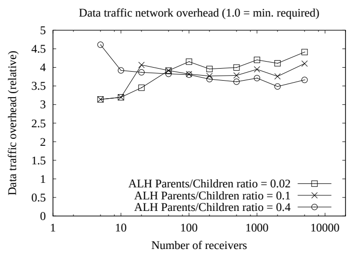

Fig. 16. ALH: network data overhead sensitivity to hierarchy organization

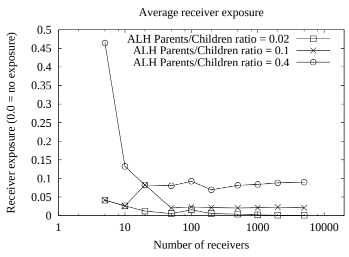

Fig. 15. ALH: exposure sensitivity to hierarchy organization

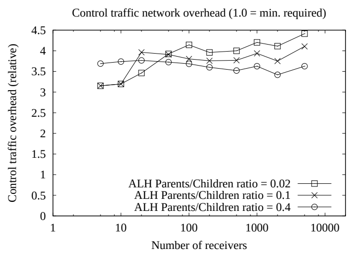

Fig. 17. ALH: network control overhead sensitivity to hierarchy organization

not have a significant impact. For ALH the control overhead was even closer to the data overhead. The reason for this can be explained by the observation that the data overhead can be smaller only if the parent used multicast, but then the gain in some parts in the network may be reduced by the exposure in other parts.

## *C. Simulation Results Sensitivity*

### C.1 ALH Hierarchy Organization Sensitivity

Figure 14 shows the latency results for three different values of the hierarchy creation parameter $frac_{pc}$: 0.02, 0.1, and 0.4 [6]. Interestingly, this parameter had almost no impact on the latency (only for a very large number of receivers the results for larger parameter value were slightly better). We believe that the reason for this is as follows. The recovery tree depth would eventually be larger when there is a large number of receivers. However, when the number of receivers increase, there is a higher probability that a parent will be on the shortest path between a child and the root (or at least almost on the shortest path). Then, if all of the parents are on the shortest path, there is no extra latency regardless of the number of intermediate hops to the root.

The data and control overhead results (Figure 16 and Figure 17) do show however, that the overhead is more sensitive to the number of parents a child has to choose from. The higher sensitivity of the data and control overhead compared to the latency sensitivity can be explained by the fact that there is a large number of leaf links (*i.e.,* when the size of the subtree that lost the data is 1), and in all those cases the overhead is much more sensitive to the distance to the parent that eventually has the data. On the contrary, the number of receivers that have very small round-trip time (the basic for comparing the latency), and therefore the distance to their parents may have a larger impact on the result, is much smaller.

From Figure 15 we can see that exposure increases when the number of potential parents is larger. The reason for the increase is because of the increased locality among all siblings, and therefore there is a larger probability the parent needs to use multicast to recover the data.

### C.2 Network Topology Sensitivity

Figure 18, Figure 19, Figure 20 and Figure 21 show the latency, receiver exposure, data overhead and control overhead results for the Mbone topology. If we compare them with the Internet-core results from Figure 10, Figure 11, Figure 12 and Figure 13 we can see that the RAH Mbone results match them closely.

The ALH-heuristic and ALH-random latency is also a reasonable match (See Figure 10 and Figure 18). However, the ALH exposure for Mbone is nearly twice higher than that for the Internet-core (see Figure 11 and Figure 19); further, the ALH-random results for Mbone do not seem to converge for a large number of receivers. This result suggests that there are probably some topology characteristics that have impact on the receiver exposure, but those characteristics for the Mbone topology do not match the Internet-map, a hypothesis which we plan to verify and explore in the future.

Finally, the data and control overhead results for Mbone (see Figure 20 and Figure 21) also seem to match the Internet-core results (see Figure 12 and Figure 13). Further, because of the relatively larger number of receivers (compared to topology size), it is clearer that the overhead for ALH-heuristic slightly increases for a larger number of receivers.

The results from our simulations did show that ALH schemes with a good hierarchy organization can perform within a constant factor of RAH schemes. Further, the ALH performance was not affected by the levels in the hierarchy, but primarily by

[6] Note that for a very small number of receivers and a small parameter value the results are identical simply because the result is always a two-level hierarchy: the sender is the root and all receivers are its children.

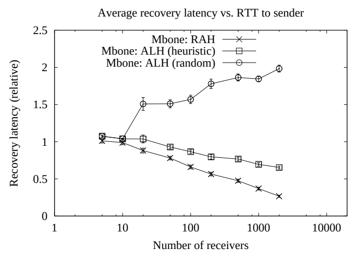

Fig. 18. RAH and ALH topology sensitivity: average latency

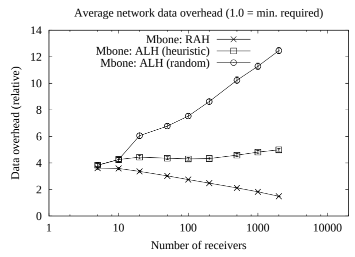

Fig. 20. RAH and ALH topology sensitivity: average network data overhead

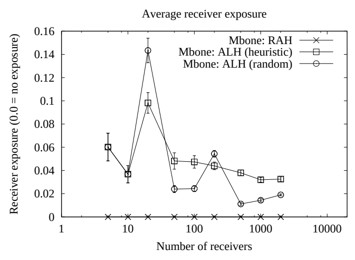

Fig. 19. RAH and ALH topology sensitivity: average exposure

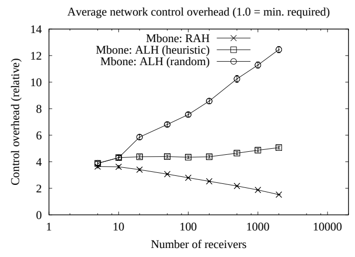

Fig. 21. RAH and ALH topology sensitivity: average network control overhead

the parent selection at each level of the hierarchy. The results were similar for both Internet core and Mbone topologies.

## V. Related Work

Previous comparisons between assisted and non-assisted schemes [14] were limited in scope compared to our current work. For example, network overhead was not considered, and the topologies used were much smaller (approximately 200 nodes) generated topologies, where here we use large (over 50K nodes) real network topologies. Moreover, in this work we have added analysis to complement our results.

We now briefly describe some of the latest proposals for reliable multicast and point out relations to our ALH and RAH schemes. While our list is not exhaustive, it covers a broad range of current proposals. We begin with the non-assisted schemes first.

SRM [3] employs two global mechanisms to limit the number of messages generated, namely duplicate suppression and backoff timers. In SRM, recovery messages (requests and replies) are multicast to the entire group; receivers listen for recovery messages from other receivers before sending their own, and suppress duplicates. Thus, SRM creates a virtual hierarchy on the fly every time there is loss in the group. However, lack of scoping means that requests and retransmissions generated by SRM will reach the entire group. Local recovery methods have been proposed for SRM [12], which bring SRM closer to our ALH scheme.

RMTP [11] is a typical example of a static hierarchical scheme which closely resembles our generic ALH scheme. The group is manually configured into Designated Receivers (DRs) and their children. DRs and their children form local groups. The source multicasts data to all receivers on the global group, but only the DRs return acknowledgments. Children unicast acknowledgments to their DRs, which schedule retransmissions using either unicast or local multicast depending on how many requests a DR has received. The Log-Based Receiver-reliable Multicast (LBRRM) [6] is another example of a static hierarchical scheme.

The Tree-based Multicast Transport Protocol (TMTP) [17] is another example of an ALH scheme, but uses a dynamic hierarchy. In TMTP, new members discover parents using an expanding ring search. Each endpoint maintains the hop distance to its parent, and each parent maintains the hop distance to its farthest child. These values are used to set the TTL field on requests and replies to limit their scope. LGMP [5] is another hierarchical, subgroup-based protocol, where receivers dynamically organize themselves into subgroups by selecting a Group Controller to coordinate local retransmissions and process feedback messages. TRAM [2] is another dynamic tree-based protocol designed to support bulk data transfer. The tree formation and maintenance algorithms borrow from other schemes like TMTP, but TRAM has a richer tree management framework. TRAM supports member repair and monitoring, pruning of unsuitable members, and aggregation and propagation of protocol related information.

Moving to router-assisted schemes, Addressable Internet Multicast (AIM) [8] is a scheme that uses forwarding services that require routers to assign per-multicast group labels to all routers participating in that group. AIM uses these labels to send a request towards the source which get redirected to the nearest upstream member. If data is available, the NACK receiver responds with a retransmission which is also forwarded according to the router labels. Active Error Recovery (AER) [7] is another scheme that is very similar to our RAH scheme. In AER, each router that has a repair server attached periodically announces its existence to the downstream routers and receivers, and serves as a retransmitter of the lost data on the subtree below it, or collects and send NACKs upstream. OTERS [10], uses a modified version of the mtrace [1] utility to build the hierarchy by incrementally identifying sub-roots using back-tracing. For each subroot, OTERS selects a parent. Unlike our RAH scheme, OTERS assumes the responsibility of discovering the topology and keeping track of changes in the structure of the underlying multicast group. Similar to OTERS, Tracer [9] also uses mtrace to allow each receiver to discover its path to the source. Once the path is discovered, receivers advertise their paths to near-by receivers using expanding ring search. Once receivers discover nearby receivers, they use the data from the traces and their loss rate to select parents.

Finally, PGM [15], unlike the schemes described earlier, peeks into transport headers to filter messages. NACKs create state at the routers which is used to suppress duplicate NACKs and guide retransmissions to receivers that requested them. PGM creates a hierarchy rooted at the source, but provision is made for suitable receivers to act as Designated Local Retransmiters (DLRs) if desired.

## VI. Conclusions

In this paper, we took a first cut at understanding the larger design question: Do router-assist schemes really perform better at error-recovery than application-level hierarchies? Our findings surprised us in many ways. First, although there was a constant factor difference in performance, the performance of ALH did not degrade with increasing group size. Second, although analysis on regular trees predicted a logarithmically increasing average latency for ALH, that trend disappeared in our simulations using irregular trees on real-world topologies. Finally, in our simulations, ALH was relatively insensitive to the depth of the recovery hierarchy.

One possible explanation for our findings is the congruence, in real and irregular networks, between a well-constructed application-level hierarchy and a router-assisted hierarchy. In this scenario, then, the performance differences arise entirely from the retransmission mechanism employed (directed subcast vis-a-vis unicast). But, in a near-optimal application-level hierarchy, the distance between parent and child is minimized, and the impact of the retransmission mechanism is small. The difference between the two schemes is significant only when losses occur near the root, and the number of levels in the hierarchy is large.

From the surprising finding that the performance difference between RAH and ALH was smaller than we had expected, it is probably too speculative to conclude that application-level hierarchies are more viable than router-assisted hierarchies for loss recovery. Our evaluations do not model the complexity and cost of hierarchy construction, particularly in the face of dynamics. They do not also consider that router-assist greatly simplifies application development.

## Acknowledgments

We would like to thank Hongsuda Tangmunarunkit and the anonymous reviewers for their valuable comments.

## References

[1] Steve Casner and Ajit Thyagarajan. *mtrace(8): Tool to print multicast path from a source to a receiver.* UNIX manual page.

[2] D. Chiu, S. Hurst, M. Kadansky, and J. Wesley. TRAM: A Tree-based Reliable Multicast Protocol. Technical Report Sun Technical Report SML TR-98-66, Sun Microsystems, July 1998.

[3] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing. *IEEE/ACM Transactions on Networking*, November 1997.

[4] Ramesh Govindan and Hongsuda Tangmunarunkit. Heuristics for Internet Map Discovery. In *Proceedings of the IEEE Infocom 2000*, Tel-Aviv, Israel, March 2000.

[5] M. Hofmann. Home page of the Local Group Concept (LGC). http://www.telematik.informatik.uni-karlsruhe.de/˜hofmann/lgc/.

[6] H. Holbrook, S. Singhal, and D. Cheriton. Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. In *Proceedings of the ACM SIGCOMM'95*, pages 328–341, Cambridge, MA, USA, August 1995.

[7] Sneha K. Kasera, Supratik Bhattacharyya, Mark Keaton, Diane Kiwior, Jim Kurose, Don Towsley, and Steve Zabele. Scalable Fair Reliable Multicast Using Active Services. *IEEE Network Magazine (Special Issue on Multicast)*, January/February 2000.

[8] B. Levine and J. J. Garcia-Luna-Aceves. Improving Internet Multicast with Routing Labels. In *Proceedings of the 5th IEEE International Conference on Network Protocols (ICNP'97)*, Atlanta, GA, USA, October 1997.

[9] Brian Neil Levine, Sanjoy Paul, and J. J. Garcia-Luna-Aceves. Organizing Multicast Receivers Deterministically According to Packet-Loss Correlation. In *Proceedings of the 6th ACM International Conference on Multimedia*, pages 201–210, September 1998.

[10] D. Li and D. R. Cheriton. OTERS (On-Tree Efficient Recovery using Subcasting): A Reliable Multicast Protocol. In *Proceedings of the 6th IEEE International Conference on Network Protocols (ICNP'98)*, pages 237–245, October 1998.

[11] J. Lin and S. Paul. RMTP: A Reliable Multicast Transport Protocol. In *Proceedings of the IEEE Infocom'96*, pages 1414–1424, San Francisco, USA, March 1996.

[12] Ching-Gung Liu, Deborah Estrin, Scott Shenker, and Lixia Zhang. Local Error Recovery in SRM: Comparison of Two Approaches. Technical Report 99-648, Department of Computer Science, University of Southern California, January 1997.

[13] Michael R. Macedonia and Donald P. Brutzman. MBone Provides Audio and Video Across the Internet. *IEEE Computer*, April 1994.

[14] Christos Papadopoulos, Guru Parulkar, and George Varghese. An Error Control Scheme for Large-Scale Multicast Applications. In *Proceedings of the IEEE Infocom'98*, pages 1188–1196, San Francisco, USA, March 1998.

[15] Tony Speakman, Dino Farinacci, Steven Lin, Alex Tweedly, Nidhi Bhaskar, Richard Edmonstone, Kelly Morse Johnson, Rajitha Sumanasekera, Lorenzo Vicisano, Jon Crowcroft, Jim Gemmell, Dan Leshchiner, Michael Luby, Todd L. Montgomery, and Luigi Rizzo. PGM Reliable Transport Protocol Specification. *Internet Draft, draft-speakman-pgm-spec-05.txt*, November 2000. Work in progress.

[16] USC/ISI. The SCAN Project. http://www.isi.edu/scan/mbone.html.

[17] Rajendra Yavatkar, James Griffoen, and Madhu Sudan. A Reliable Dissemination Protocol for Interactive Collaborative Applications. In *Proceedings of the Third International Conference on Multimedia '95*, San Francisco, CA, USA, November 1995.