An Error Control Scheme for Large-Scale Multicast Applications

Christos Papadopoulos
christos@dworkin.wustl.edu
Guru Parulkar
guru@arl.wustl.edu
George Varghese
varghese@askew.wustl.edu
Washington University,
St. Louis, MO

ABSTRACT

Retransmission based error control for large scale multicast applications is difficult because of implosion and exposure. Existing schemes (SRM, RMTP, TMTP, LBRRM) have good solutions to implosion, but only approximate solutions to exposure. We present a scheme that achieves finer grain fault recovery by exploiting new forwarding services that allow us to create a dynamic hierarchy of receivers. We extend the IP Multicast service model so that routers provide a more refined form of multicasting (which may be useful to other applications), that enables local recovery. The new services are simple to implement and do not require routers to examine or store application packets; hence, they do not violate layering. Besides providing better implosion control and less exposure than other schemes, our scheme integrates well with the current IP model, has small recovery latencies (it requires no back-off delays), and completely isolates group members from topology. Our scheme can be used with a variety of multicast routing protocols, including DVMRP and PIM. We have implemented our scheme in NetBSD Unix, using about 250 lines of new C-code. The implementation requires two new IP options, 4 additional bytes in each routing entry and a slight modification to IGMP reports. The forwarding overhead incurred by the new services is actually lower than forwarding normal multicast traffic.

Key words: reliable multicast, error control

1. INTRODUCTION[1] [2]

The recent growth of the MBONE and other multicast-capable networks has led to the widespread deployment of multicast applications such as video-conferencing, distributed interactive simulation, and news distribution. Many of these applications require data delivery guarantees not provided by IP Multicast [1]. Thus, we require multicast transport protocols that work on top of the network multicast service to provide delivery guarantees.

Large scale applications (e.g., DIS, bulk data distribution) complicate the problem. These applications have large numbers (hundreds or even thousands) of participants who may be distributed worldwide. In addition, the highly dynamic nature of the topology and population poses new, difficult challenges to traditional error control schemes. Such control schemes (e.g., TCP [9]), used primarily in point-to-point applications, do not scale to meet the demands of large-scale multicast. The difficulties arise because of the IP Multicast service model, which requires that all messages sent to a multicast address reach all receivers subscribing to that address. While this model is simple, elegant and works well for data transfer, it is not suitable for data recovery because losses in a multicast environment are local, i.e., they affect only part of the multicast tree. Attempting to recover data lost locally by global means leads to exposure, since receivers have no choice but to multicast requests and retransmissions to the entire group. As the group size gets large, exposure leads to implosion, due to the multicast of numerous requests and replies.

Currently proposed approaches that deal with implosion and exposure within the context of the current IP service model have done so at the expense of some other performance aspect, e.g., latency or topology-related state and processing. Latency is not tolerated by some applications; topology related state and processing makes it hard for groups to adapt to dynamic membership and topology changes.

It is our belief that the following problems should be addressed in reliable multicast:

1. **Implosion**: a problem that occurs when the loss of a packet triggers simultaneous messages (requests and/or replies) from a large number of receivers.
2. **Exposure**: a problem that occurs when recovery-related messages reach receivers which have not experienced loss.
3. **Recovery latency**: the latency experienced by a member from the instant a loss is detected until a reply is received.
4. **Adaptability** to dynamic membership changes: a measure of how the efficiency (in terms of loss of service, duplicate messages and added processing and/or latency) of error recovery is affected by changes in the group topology and membership.

We have designed, simulated and are implementing in NetBSD Unix a multicast error control scheme that addresses the problems enumerated above, and offers significant improvements over existing schemes. Our scheme extends the IP Multicast service model to allow routers to offer a small set of new forwarding services. Applications develop error control schemes that leverage off these services to provide reliability. The implementation of these services at the routers eliminates the need for endpoints to learn about group topology, but allows easy access to topology information, which leads to an efficient implementation. The services are conceptually simple, do not violate the end-to-end argument and have no effect on current IP routing protocols.

[1]Supported in part by NTT, Tektronix, NIH, NEC, SWB, TRI, Samsung and Sun. George Varghese supported by NSF Research grant NCR-9612853.
[2]An expanded version of this paper can be found in [6]

This paper is structured as follows. In Section 2, we summarize related work; in Section 3, we present an overview of our scheme; in Section 4, we present the mechanisms in more detail and discuss some limitations of our scheme; in Section 5, we present numerical and simulation results; in Section 6, we discuss our implementation in NetBSD Unix; and finally, in Section 7, we conclude this report.

2. RELATED WORK

Several solutions have been proposed to deal with the problems enumerated above, within the existing IP service model. SRM[3] uses two clever global mechanisms to limit the number of messages generated, namely duplicate suppression and back-off timers. In SRM, recovery messages are multicast to the entire group; receivers listen for recovery messages from other receivers before sending their own, and suppress their messages if they would duplicate one already seen. The intended goal is to allow the multicast of only one message. In order to increase the effectiveness of the suppression mechanism, especially in densely packed groups, the round-trip-time between receivers is artificially enlarged (for recovery messages only) with the addition of back-off delay. To handle both linear and star topologies, the added delay consists of a fixed and a random component, calculated separately at each receiver. The fixed component is based on the distance of the receiver from the sender, and the random component is based on the density of the receiver neighborhood. Thus, these components have to be re-calculated when group membership, topology, or network conditions change, meaning that SRM takes time to adapt to offer its best performance. Finally, to limit exposure, SRM limits the scope of recovery messages to a radius using the TTL field in the IP header. However, estimating the appropriate TTL value is hard, and is at best a crude method of controlling exposure. Thus, while SRM is a conceptually elegant scheme and offers good robustness, its performance and scalability suffer as a result of using global mechanisms to solve local problems. While SRM preserves the simplicity of the current multicast model, its randomization and estimation algorithms are quite complex, and we believe that its trade-offs may not be appropriate for the requirements of future multicast applications.

Another way to achieve local recovery is to create separate recovery groups for requests and replies. Creating a group dynamically for every loss is too costly and slow; therefore, it is better to precompute such groups for each loss region, or create them on demand as the group topology changes. However, creating a new group is a slow process and requires prune messages to propagate throughout the entire network. In groups with highly dynamic membership, such a scheme will incur significant overhead.

Other approaches adopt different methods to control implosion and exposure. These are called hierarchical, or tree-based approaches, named after organizing receivers in a recovery tree. Such approaches include, RMTP[7], LBRRM[4] and TMTP[12]. Exposure and implosion are limited by allowing recovery messages to traverse only between parents and their children. Parents and children must keep state about each other, which has to be updated as the group topology changes. In addition, the success of such approaches depends on the accuracy with which the recovery tree follows the underlying routing tree. However, it is hard to create a recovery tree that faithfully follows the routing tree without topology information. Therefore, some approaches rely on the construction of static recovery trees (RMTP, LBRRM), which limits adaptability, while others (TMTP) have to employ distributed algorithms based on heuristics to estimate the topology of the routing tree. However, the latter is only approximate and potentially expensive, especially if group membership and topology change frequently.

Other schemes that require modifications to the routers have been recently proposed. Search Party [2] is very similar to our scheme, except that it uses random routing of requests to enhance robustness at the expense of added latency. In addition, it distributes the load of sending retransmissions to many members instead of a single replier. AIM[5] assigns group-relative routing and distance labels to the routers in a multicast group, which enable positional routing within the group. A Reliable Multicast Architecture (RMA) is built based on these services. AIM works well with routing protocols like CBT, where our scheme in its current form has difficulties. However, AIM requires substantial changes to the routers and the overhead involved, like distributing and updating positional and distance labels, are much higher than in our scheme, especially in the presence of highly dynamic groups.

3. SOLUTION OVERVIEW

In a multicast environment, loss of a packet in the network results in failure to deliver a copy of the packet to all receivers located in the subtree rooted at the branch sprigging from the point of loss, as shown in Fig. 1(a). A natural solution to recover the packet is the following (Fig. 1(b)):

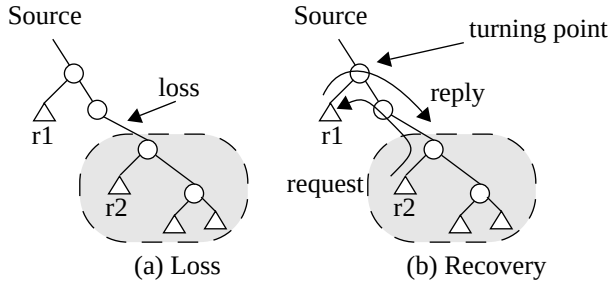


Figure 1: Recovery steps with topology knowledge

1. the receiver directly below the loss (r2) sends a request to the receiver immediately above the loss (r1)
2. r1 multicasts the lost packet to the affected branch.

Note that both steps require topology information. We claim that this solution is optimal because, (a) only one request and only one reply are generated, (b) replies are visible only within the affected branch, and (c) requests and replies traverse the shortest possible distance. We attempt to duplicate these steps in our scheme. Since topology information is required, we turn to the network for help. Routers, with their knowledge of topology, are the ideal candidates to provide services to navigate a request upstream to a willing replier, and help the replier multicast replies to the loss subtree.

It is important to note that such services can be implemented

without requiring routers to be aware of error recovery. Routers do not have to keep state, like sequence numbers, or provide any guarantees to endpoints using the services. Routers simply provide the means for receivers to reach other receivers in a manner that happens to be useful for error recovery. Receivers are free to build their own recovery mechanisms on top of these services. We believe that such services do not violate the end-to-end argument if they are pure forwarding services and do not require packet examination. Moreover, we believe that adding such services to the network is justified if they result in more efficient error recovery than currently proposed schemes, provided they do not introduce unacceptable overhead in the network.

To realize these services we introduce three new concepts. First, we use routing to calculate a **replier** for each subtree. We use the replier to respond to requests made by other endnodes in the replier subtree, thus creating a hierarchy. Second, we use routing to define a **turning point**. A turning point is the point in the topology at which a request moving upstream is moved downward towards the replier (see Fig. 1). When a replier wants to send a retransmission, the replier contacts the router at the turning point to perform a **directed multicast** to deliver the retransmission to the loss subtree. With these three concepts (electing repliers, defining turning points, and using directed multicast for retransmissions), we closely approximate the optimal recovery steps depicted in Fig. 1.

It is important to quickly see why these three ideas help our scheme do well with respect to the measures described in the Introduction. The use of a replier hierarchy prevents request implosion; reply implosion is eliminated by allowing only one replier to respond. The scheme has optimal recovery latency, because no backoff delays are needed and replies come from the closest replier. The use of turning points and directed multicasts help isolate recovery to the subtree affected by the fault. Finally, membership and topology changes are easily adapted to by routers as group membership changes by calculating new multicast trees and repliers for each subtree, if needed, without endnode involvement.

We now present an overview of these three major steps. Section 4 contains additional details

3.1 Adding a Replier to each Router

Each router selects a *replier link* for every (S, G) routing entry. The replier is selected from the router's downstream links; the only exceptions are routers with only one downstream link, which select the upstream link as the replier link, and the router adjacent to the source, which selects the source link as its replier link. The upstream/downstream link information is maintained by the multicast routing protocol[3]. To aid in replier selection, receivers notify the routers when they join or refresh their membership in the multicast group; that is, along with the join or refresh message, a receiver indicates its willingness to act as a replier. Whenever its replier state changes (e.g., as a result of a membership change), a router propagates the change upstream so that other routers can update their replier links. Thus, the maintenance of replier state incurs minimal overhead and is automatically updated as the group membership changes.

[3]Most popular multicast routing protocols, including DVMRP and MOSPF, maintain per-source routing state; but see discussion on PIM-SM later.

3.2 Step 1: Sending Requests

With all repliers in place, sending a request is done as follows: after detecting a loss (e.g., a gap), a receiver sends a request to the router. The router forwards all requests to the replier link, except the one from the replier link, which is forwarded upstream. Thus, at most one request is sent upstream, reducing request implosion. In addition, requests travel upstream only until they reach the next replier, reducing exposure.

These operations are shown in Fig. 2(a). Replier links are in

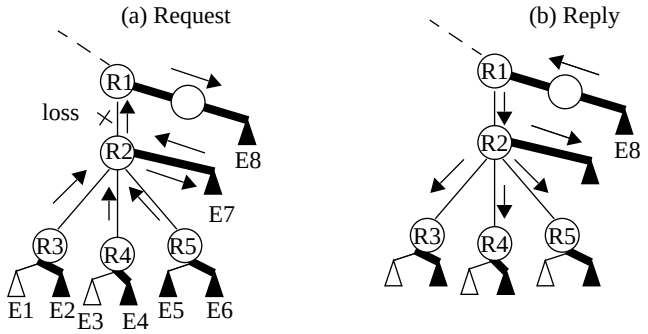


Figure 2: Request - Reply operations for error recovery

bold. Loss occurs on the link between R1 and R2. Endpoints E1 through E7 detect the loss. Then, the following events take place:

- E7 sends a request, which R2 forwards to R1 because E7 lies on R2's replier link.
- E1 sends a request which is forwarded by R3 to E2. Similarly, requests from E3 and E5 are forwarded to E4 and E6 by R4 and R5, respectively.
- The request from E2 is forwarded to R2, because E2 is on R3's replier link. Similarly, the requests from E4 and E6 are also forwarded to R2.
- R2 forwards requests from E2, E4 and E6, to E7.
- The request from E7 reaches R1, which forwards it towards E8, which has the requested data.

3.3 Step 2: The Turning Point

This is an important concept in our scheme. We define the turning point in the request's path towards the replier as the router which forwards the request to a replier. Thus, in the previous Figure, the turning point for the request sent by E7 is R1, and the turning point for the requests sent by E2, E4 and E6, is R2. When a request passes through its turning point, the router inserts into the request the router's address and the identifier for the link on which the request arrived. Other routers on the path to the replier do not change this information. Thus, requests traveling downstream carry their turning point with them. Turning points define the root of the loss subtree and are used in sending replies, as described next.

3.4 Step 3: Sending Replies using a Directed Multicast (DMCAST)

If a replier receives a request but does not have the requested data, the replier ignores the request since it must have sent a similar request of its own. Otherwise, the replier retransmits the data using a DMCAST. To do so, the replier creates a reply containing the data and the link identifier at the turning point. The replier then unicasts the reply to the turning point router. When the router receives the unicast, it extracts the data and multicasts it on the specified link.

In Fig. 2(b), these steps are as follows:

- E8 creates a multicast message containing the reply. E8 encapsulates the message in a unicast message and sends it to R1 (the request's turning point).
- R1 decapsulates the multicast message and multicasts it on the link leading to R2.
- From that point on, all downstream routers and endpoints treat the reply as a regular multicast message coming from the source.

It should be clear from Fig. 2 that the turning point is the root of the subtree which has lost the requested data. Thus, establishing the turning point before multicasting a reply is a crucial step in containing replies to the loss region and allows our scheme to achieve very good isolation.

4. DESIGN DETAILS

The previous section has given an overview of how the new services can be used for error recovery. In this section, we offer details that were glossed over in the previous section.

4.1 Establishing Replier State at Routers

Following the IP multicast model, the replier state at the routers is soft state. It requires the addition of the following information to the per-source state already maintained by the routing protocol:

- a tag to identify the replier link
- the cost to reach the current replier (e.g., hop count or current replier loss)

The new state adds 4 bytes per routing entry, which is insignificant compared to the entry's current size. The purpose of *cost* is to capture differences between multiple potential repliers, so that the best one can be selected.The cost may be the loss rate experienced by the current replier so that the most reliable replier is selected, or the router-replier distance so that the closest replier is selected.

Since replier state is maintained on a per sender basis, it is possible that in the worst case a router may have to select a replier for every sender in a multicast group. However, we expect that in reality many receivers will advertise that they are willing to act as repliers for several (or maybe all) senders. A router can then select a single replier for all senders who share the same upstream link, thus significantly reducing the required replier state.

Receivers which want to act as repliers, send periodic messages to the routers, advertising a cost to help the router select the replier with the least cost. Routers examine the cost advertised by the message; if the message came from the replier link and the new cost is equal to the current cost, the router refreshes the replier expiration timer. Otherwise, if the new cost is less than the current cost, the old cost is discarded and the new value is stored. If the link the message was received on is different than the current replier link, the router updates its replier link. The replier expiration timer is then reset.

Whenever a router's replier state changes, the router propagates the change to the upstream routers, which repeat the same operations. Thus, replier state propagates to all routers on the multicast tree. As an optimization, it is beneficial (but not required) that routers cache the next best replier link so they can instantly switch to the cached link if the current replier leaves the group or fails.

Note that a router is not required to store the replier's address because a replier does not have to be notified when it is selected by a router. Similarly, upstream routers do not have to notify downstream routers if they select them as part of the replier path. This saves complexity and state, and allows routers the flexibility of switching repliers at will.

The mechanism for selecting repliers can be easily integrated with the group membership protocol (e.g., IGMP) so that the replier state is created while the group is being formed. For example, when a member joins or refreshes its membership in a group, it may also refresh its replier status with the same message. Thus, very little work is needed for creating and maintaining the replier state. In return, the replier state is updated instantly as the group grows or shrinks.

4.2 Sending Retransmission Requests

When a loss occurs, receivers must send a request which the routers will deliver to an upstream replier. To do so, we create a new type of control message to carry retransmission requests, which is examined by every router in its path. Routers identify this control message via a hop-by-hop option in the multicast header. The control message is multicast to the group in the normal fashion.

The control information of the message contains two items. The first item is the <source, multicast address> for the group. This information is required to identify the appropriate source tree when routing the message to a replier. The second control item is a <router address, link identifier> entry, used to mark the turning point. This entry is initially empty.

At the routers, the following actions are taken:

- If the message came from a downstream link and either (a) the router has no replier link, or (b) the message came from the replier link, then the router forwards the message on the upstream link leaving the control information unchanged.
- If the message came from a downstream link other than the replier link, then this is the turning point. The router fills in the <router addr, link id> fields and forwards the message on the replier link.
- If the message came from the upstream link and (a) the <router addr, link id> fields are not empty, and (b) a replier link exists, the router forwards the message to the replier link unchanged. If the <router addr, link id> fields are empty, the message is silently discarded. If no replier link exists, the router signals an

error (some upstream router erroneously thinks that this path leads to a replier).

Requests must be examined by all routers along their path due to their non-standard forwarding. However, we believe that integrating the request processing in the fast path of a router is feasible because the overhead for forwarding these messages is very low. This information is made part of the routing record, so that it is readily available once the routing lookup is performed. The remaining operations require on the order of 10 instructions or less. Only the router at the turning point has to actually touch the header, which may require another 10 instructions or so.

4.3 Sending Replies using DMCAST

The directed multicast service is a crucial element in preventing exposure. A directed multicast consists of two parts: a unicast from the replier to the router, and a subsequent multicast by the router on one of the router's links. To perform a directed multicast, the replier creates a multicast packet, encapsulates it in a unicast packet and sends it to the router. In the unicast, the replier specifies which of the router's links the packet should be multicast. The router decapsulates the packet, performs some validity checks (see below) and multicasts it on the requested link. A directed multicast can be summarized as follows:

1. A replier receives a request, which contains a list of lost packets and the turning point information.
2. If the replier has the requested data, the replier creates a multicast packet containing the reply. The multicast packet is then unicast to the router at the turning point.
3. The router decapsulates the multicast packet and checks the validity of the group and link specified in the message. If valid, the router multicasts the packet on the specified downstream link.

Note that in a directed multicast only the router at the turning point performs operations beyond normal forwarding and the overhead is comparable to the overhead required to forward a regular multicast packet.

4.4 Source Spoofing

Some routing protocols (e.g., DVMRP) create a separate multicast tree for each sender. With such protocols, the multicast reply resulting from a directed multicast must contain the original sender's address as the source address, otherwise it will not reach the appropriate receivers. To avoid this problem, we allow repliers to use the original source's address in the multicast packet (i.e., perform *"source spoofing"*). However, to allow receivers to distinguish spoofed from real packets, routers ensure that spoofed packets are marked and include the replier's address in the message. Thus, source spoofing poses no additional security concerns since the real sender can always be identified by the recipient. Source spoofing is unnecessary with routing protocols that create shared trees.

4.5 Duplicate Data Packets

Since there is always one replier in our scheme, no duplicate replies are ever generated. It is possible, however, that a receiver may be exposed to replies generated as a result of recovery initiated by receivers in other regions.

An example of such exposure is depicted in Fig. 3. In this

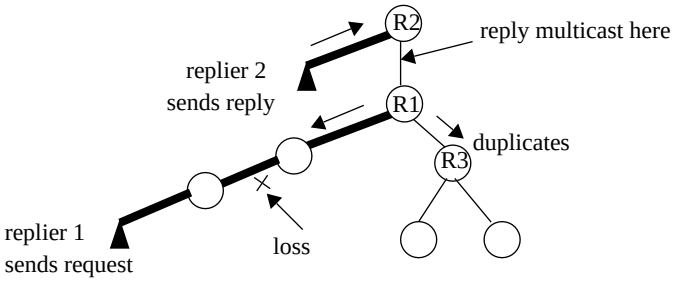


Figure 3: Loss on replier path causes duplicate messages

example, a packet is lost on the path between R1 and replier 1. Replier 1 sends a request which reaches replier 2. In response to the request, replier 2 sends a directed multicast to R2, which multicasts the reply on the downstream link leading to R1. The reply reaches all of R1's downstream links, causing duplicates on the subtree routed at R3.

Even though this problem does not inhibit recovery, it may lead to the "crying baby problem," where excessive loss experienced in one branch causes duplicates at a large number of other receivers. This problem is solved by using the cost field to select a replier that advertises the least loss. For example, R1 will select a replier from the right-hand-side branch if this branch experiences less loss, even though the replier on the left-hand-side branch may be closer.

4.6 Dealing with Lack of Per-source Routing State

While most of the multicast capable routers on the MBONE today use DVMRP, some new routing protocols like PIM-SM and CBT create shared multicast trees around a *core* or *rendezvous point*, and thus do not maintain per-source information. This enables them to scale to groups with many sources. In order to handle these protocols, we propose the following changes to our scheme, as depicted in Fig. 4:

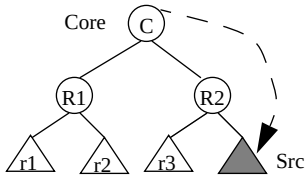


Figure 4: Dealing with shared trees

We calculate subtree leaders (repliers) as before for the core based tree. A request is directed by the routers to the leaders as before. Requests from the leaders are directed towards the core. In order to guarantee that the source will eventually receive the request, whenever the core receives a request it unicasts it to the source. The source in turn performs a directed multicast to the turning point as before. So rather than have the source directly connected to the root as in DVMRP schemes, the source is con-

nected by a unicast path to the root.

The above modification works well for PIM-SM. Sources in PIM-SM initially send data to the core via register messages, which are then multicast by the core. Thus the core always lies upstream. If the core sends a join message to the source, then intermediate routers along the join path maintain per-source information, which allows correct routing of requests. However, in CBT, routers can no longer distinguish upstream and downstream links with respect to a source. This may result to misdirection of requests and some loss of isolation.

4.7 Selecting Repliers in a LAN

For simplicity, the previous sections have assumed that only one receiver resides at each router link. It is easy, however, to coordinate receivers on a LAN to behave as a single host, by electing a replier among them. The election does not require any involvement from the router. See [6] for more details.

4.8 Replier Failure

The failure of a replier may disrupt recovery until soft state expires and the failed replier is detected. Note that a failed replier will not always disrupt recovery; replier failure becomes problematic only if the failed replier is located directly above or below the loss (see Fig. 1). If the failed replier is located anywhere else, recovery will proceed unaffected. Soft state allows routers to eventually detect failed repliers. However, detection via soft state may take too long; to enable fast detection of replier failure, a receiver may send a probe message which the replier must immediately acknowledge. If no acknowledgment is received, the receiver sends another message to the router at the turning point, alerting the router of possible replier failure.

5. RESULTS

As mentioned earlier, request implosion is controlled by using the replier hierarchy. There are no duplicate replies in our scheme because there is always one replier. We have not yet evaluated the adaptability of our scheme to topology changes; however, we expect it to be very good since the replier state changes as group membership changes.

In this section we investigate the remaining two problems, namely exposure and latency, with simple numerical analysis and simulation.

5.1 Numeric Results: Exposure

We define exposure as:

$$Exposure = \frac{\#\ receivers\ that\ received\ a\ reply}{\#\ receivers\ that\ should\ have\ received\ the\ reply}$$

We calculated the exposure in our scheme and compared it to a a global scheme without local recovery. The exposure depends heavily on topology because the location of loss determines which receivers miss the packet. We chose a binary tree as our target topology. Exposure was then calculated as follows:

- drop a packet on a link at height *h*
- calculate the resulting exposure due to that fault
- repeat the above until a packet was dropped on all links of height *h*
- report average exposure

Table 1: Exposure with a binary tree topology

Height	Without Local Recovery	Our Scheme
1	2	1
2	4	1.5
3	8	2
4	16	2.5
...	...	...
h	2^h	O(h)

The results are presented in Table 1. From the table, we see that without local recovery the exposure increases exponentially as losses move closer to the leaves. In contrast, with the local recovery offered with our scheme, exposure increases only linearly. Note that recent studies of losses on the MBONE indicate that most losses tend to occur at the leaves [11].

5.2 Simulation Results

We simulated our scheme and measured *Nuisance* and the recovery latency. We defined nuisance as follows:

$$Nuisance = \frac{\#\ unwanted\ replies\ received\ by\ R}{total\ number\ of\ losses}$$

Thus, schemes without local recovery have nuisance = 1. Recall that since in our scheme there is only one replier, a duplicate is an unwanted message received as a result of recovery elsewhere. Thus, in contrast with exposure, which measures how many receivers were exposed to recovery, nuisance measures how often a receiver is pestered by messages from recovery elsewhere. In our simulation we simulated the following:

- A single multicast tree with one sender.
- All the router functionality, including replier setup, forwarding messages to repliers, the turning point and directed multicasts. The cost used by routers to select repliers is the hop distance.
- Gap-based error detection at the receivers. All receivers participate in error recovery.
- Loss of original data packets only. Requests, retransmissions, and other control messages are not lost.

The inputs to the simulation are the topology, link parameters (bandwidth, latency, loss) and the type of loss to be simulated (random or deterministic). The outputs were the average recovery latency and the average nuisance. Runs were made on various topologies. We present results from two types of topologies: binary trees and a WAN-like topology.

5.2.1 Simulations with Binary Trees

Binary trees, although not a very realistic topology, are useful because they provide a controlled environment. Binary trees actu-

ally represent a difficult case for our scheme because the lack of internal repliers increases the replier paths, which increases duplicates and latency.

We assigned all links identical bandwidth, latency, and loss probability, simulated binary tree topologies of various heights. The results were obtained as follows: a packet was dropped on a link; the unwanted replies (if any) and recovery latency were measured at all receivers; a packet was then dropped on a different link and the measurements were repeated, until one packet was dropped on every link. The above is equivalent to each link having equal loss probability. The results were then averaged over all receivers to calculate the average nuisance and average latency. The normalized latency is obtained by dividing the measured latency with the RTT between the receiver and the sender.

Table 2 shows the results. Average latency remains under unicast RTT. Nuisance is low, despite the fact that there are no "good" repliers in these topologies.

Table 2: Nuisance and latency for binary trees

Tree Height	Nuisance	Avg normalized latency (avg latency / RTT)
3 (8 receivers)	0.11	0.92
4 (16 receivers)	0.10	0.93
5 (32 receivers)	0.08	0.95
6 (64 receivers)	0.06	0.96

Adding receivers to the internal routers can only improve performance in our scheme since recovery messages need to travel shorter distances and directed multicasts can be more accurately aimed at the loss region. Thus, we examine what happens if we add more receivers at the leaves. This better approximates real topologies, where receivers are typically concentrated at the edges. The simulation is run as before, and Table 3 shows the results. The original result (2 receivers per leaf) is taken directly from Table 2. The table shows that adding more receivers at the edges decreases both nuisance and latency. This was expected, because adding receivers where a replier already exists does not increase exposure; on the contrary, it allows new receivers to recover faster and with less exposure.

Table 3: Nuisance and latency for different receivers per leaf router

Tree height	Receivers per leaf router	Nuisance	Avg normalized latency (avg latency / RTT)
3	2	0.11	0.92
3	3	0.09	0.889
3	4	0.08	0.875
4	2	0.1	0.93
4	3	0.08	0.908
4	4	0.07	0.897

5.2.2 WAN simulation

For our final set of results, we simulated the WAN topology depicted in Fig. 5. The topology is imaginary, but attempts to capture some elements of a typical WAN. There are 14 routers and 33 receivers. The selected repliers and replier links are shown in bold. The propagation delay between any router and a receiver is assumed to be 1 ms, a typical value in a LAN. The simulation is performed as before, by dropping one packet on each link, and the results are shown on Table 4. The second row shows how the performance changes if the source is moved to R3, bringing it closer to the "middle" of the multicast tree. Note that recovery latency has been reduced almost by half.

Table 4: WAN

WAN: 14 routers, 33 receivers	Nuisance	Avg normalized latency (avg latency / RTT)
Source at R0	0.1	0.6
Source at R3	0.04	0.49

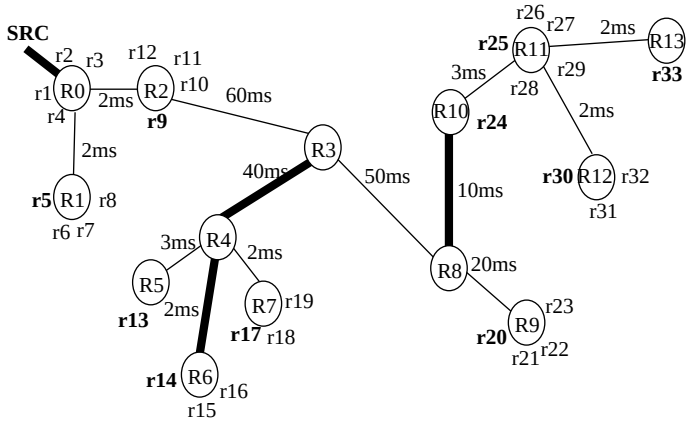


Figure 5: Imaginary WAN topology

6. IMPLEMENTATION

We have modified the NetBSD Unix kernel to support handling of requests and directed multicasts. The kernel modifications include the addition of two new IP multicast options, namely IPOPT_MREQ and IPOPT_DMCAST. Requests carry the IPOPT_MREQ option in the IP header; replies are encapsulated in unicast packets which carry the IPOPT_DMCAST option. Control information from the user is conveyed to the protocol as *ancillary data* (see below). The modified kernel components are: UDP output processing, IP and UDP input processing, and kernel multicast forwarding. We added about 250 lines of new C-code to the kernel; interested readers can peruse the code at http://dworkin.wustl.edu/~christos/.

6.1 Ancillary Data in NetBSD Socket Interface

When a user sends a request or a reply, some control information must be passed to the kernel along with the data. For a

request, this includes the <*source, group*> pair; for a directed multicast, this includes the address of the router and the link at the turning point. The NetBSD socket interface provides two ways of user/kernel control information exchange. The first is via the system call setsockopt. The second is via the sendmsg and recvmsg system calls. These calls accept as arguments control parameters (called ancillary data) that are passed along with the normal data when a packet is sent, as depicted in Fig. 6. These

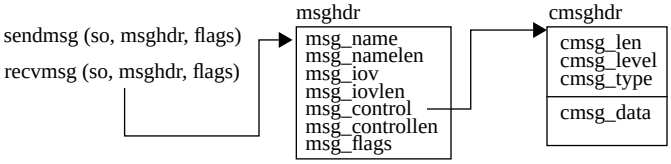


Figure 6: Ancillary data in NetBSD

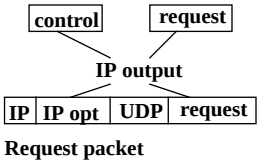
calls can be used to pass control information that affect an individual packet, rather than a stream of packets as with setsockopt, and thus are more appropriate for requests and directed multicasts. The same mechanism being considered for setting and retrieving information carried in IPv6 header options [14].

We now describe how sending and receiving requests and directed multicasts are implemented using these calls.

6.2 Sending/Receiving Requests

Sending a request is done as follows: upon detection of a gap, the application creates a retransmission request with a list of the missing packets. Then, the application allocates a cmsghdr structure, sets cmsg_type = IPOPT_MREQ, and writes the address of the original multicast source in cmsg_data. The application allocates a msghdr structure, attaches the request to the msg_iov field and the cmsghdr structure to the msg_control field and calls sendmsg to send the request.

In the kernel the call reaches UDP, where the control information is converted into an IP option. UDP then passes the packet along with the option to ip_output, as shown in Fig. 7. The packet



Send overhead	Receive overhead
20 bytes at sendmsg call to ip_pcbopts 12 byte IP option	26 bytes at recvmsg 10 bytes control data 10 byte IP option

Figure 7: Sending/receiving a request at an endpoint

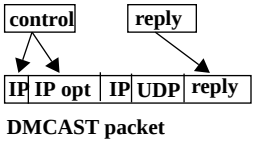
is then multicast through the normal path.

A request is received by all members on the requestor's LAN. This is desirable in order to suppress similar requests. At a router, however, a request is either forwarded upstream or to the replier link. When a request finally reaches a replier, the turning point information is extracted from the IP option by UDP and passed to sbappendaddr which appends it to the receive socket buffer as control information. The application retrieves the request and control information with a call to recvmsg.

The overhead of sending and receiving a request is summarized in Fig. 7. The table shows overhead *in addition* to the normal sendmsg/recvmsg overhead. At the sending side, the overhead consists of 20 control bytes, out of which 4 bytes are for the address of the original source and 16 bytes are for cmsghdr. This is followed by a call to ip_pcbopts to prepare the IP option which is 12 bytes (option type and length, src addr, router addr and router link). At the receiving side, 10 bytes of the IP option are copied into a cmsghdr structure and delivered to the application via the recvmsg system call.

6.3 Sending/Receiving DMCASTS

Similar to sending a request, the application creates a reply and a control message with the turning point information, and calls sendmsg to multicast the packet. UDP, however, intercepts the multicast packet and passes it to a new function, ipudp_encap. There, the packet is encapsulated in a unicast packet, whose destination is taken from the control information. The link id is copied from the control information to a new IP option, IPOPT_DMCAST. The process is depicted in Fig. 8.



Send overhead	Receive overhead
22 bytes at sendmsg call to ip_pcbopts encapsulation IP hdr 4 byte IP option	None

Figure 8: Sending a directed multicast from an endpoint

The overhead for sending a directed multicast is similar to sending a request, but with a 4-byte IP option (type, length and link index), and the IP encapsulation header. However, there is no additional overhead for receiving directed multicasts at the endpoints, as they are regular multicast packets.

6.4 Handling of Requests at a Router

In order to handle requests correctly, a router which has more than two interfaces belonging the same multicast group, must maintain a replier entry for each source. This requires the addition of two new fields to struct mfc, which is the *multicast forwarding cache* entry. The new fields are, vifi_t mfc_replier which contains the index of the replier interface, and u_short mfc_replier_cost, which contains the cost of the current replier. These fields are set by mrouted in response to IGMP messages.

When a multicast packet carrying the IPMOPT_MREQ option arrives at the router, it is passed to a new function, ip_mfwdrequest. This function performs a route lookup based on the source address carried in the option, and the multicast group address; this returns the cache entry for the original source. Then, the mfc_replier field of the entry is compared to the incoming interface. If they match, the packet is sent out to mfc_parent; otherwise it is sent out to mfc_replier. In the latter case, before forwarding the packet, ip_mfwdrequest fills the turning point information in the IP option.

6.5 Handling DMCASTs at a Router

Unlike requests, directed multicasts arrive at the router encap-

sulated in unicast packets. When such a packet arrives, it is passed to a new function, ip_dmcast. This function retrieves the interface index from the option, strips the unicast IP header and the option from the packet, and, if the interface index is valid, forwards the multicast packet out the interface.

The overhead of the new functions (ip_mfwdrequest and ip_dmcast) is lower than the regular IP multicast forwarding function, ip_mforward, because there is no need to scan all the outgoing interfaces for membership and duplicate packets.

6.6 Setting Replier Information: Modifications to IGMP

This part of the implementation has not been completed at the time of writing. Our experiments were run using a fixed replier entry. We plan to modify IGMP to carry information to aid mrouted set the replier information. We will make the source code available at our web site when ready.

7. CONCLUSIONS

In this paper we have described an error control scheme for large multicast groups, based on a new set of forwarding services provided by routers. These new services are simple to implement (can be included in the router's fast path) and easy to integrate with the current Internet multicast model, while incurring minimal overhead at the routers. In return, these services allow receivers to implement efficient and fast error recovery, while maintaining their isolation from the topology of the group. These services are very general and can be cleanly implemented without exposing routers to the details of the particular error recovery mechanism used by the receivers.

Our scheme shows significant performance gains compared to other schemes, with respect to the four metrics, as shown in Section 5. The biggest gains are in terms of lower latency and exposure. The scheme typically recovers from errors in one round-trip delay or less. The number of receivers exposed to recovery is reduced from a factor of 2^h without local recovery, to h, where h is the height of the multicast tree. Even though we did not simulate the scheme's adaptability to a changing group topology, we believe that it is very good. Our scheme requires about 250 lines of C-code in the kernel, and forwarding requests and directed multicasts requires less overhead than normal traffic.

We believe that these new services can be useful for other purposes. For example, members of a group can implement a positive acknowledgment reliable transport service by selecting a fixed set of repliers to collect acks. Other applications could also use directed multicast to provide a much tighter form of scoped multicast than what is available today. In addition, these services can be used as the basis for a congestion control scheme that gathers loss information from repliers.

Routers enhanced with our services may be deployed on the MBONE using a mechanism similar to the one used in RSVP [13], or in IPv6[15]. Note that non-enhanced portions of the multicast tree may also benefit from recovery performed by the enhanced tree. We are investigating the feasibility of such integration.

REFERENCES

[1] Deering, S., "Host Extensions for IP Multicasting," RFC 1112, January 1989.

[2] Costello, A., "Search Party: An approach to Reliable Multicast with Local Recovery", work in progress, http://www.cs.berkeley.edu/~amc/research/search-party/.

[3] Floyd, S., Jacobson, V., McCanne, S., Zhang, L., Liu, C., "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing," Proc. of ACM Sigcomm'95, pp. 342-356, September 1995.

[4] Holbrook, H., Singhal, S., Cheriton, D., "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation," Proceedings of ACM Sigcomm'95, Vol. 25, No. 4, pp. 328-341, October 1995.

[5] Levine, B., Garcia-Luna-Aceves, J.J., "Improving Internet Multicast with Routing Labels", Proc. of IEEE ICNP, Atlanta, GA, Oct. 1997, http://www.cse.ucsc.edu/research/ccrg/publications.html.

[6] Papadopoulos, C., Parulkar, G., Varghese, G., "An Error Control Scheme for Large-Scale Multicast Applications", expanded version, http://dworkin.wustl.edu/~christos/PostScriptDocs/Infocom98.ps.Z

[7] Paul, S., Sabnani, K.,. Buskens, R., Muhammad, S., Lin, J., Bhattacharyya, S., "RMTP: A Reliable Multicast Transport Protocol for High-Speed Networks," Proceedings of the Tenth Annual IEEE Workshop on Computer Communications, September 1995.

[8] Pingali, S., Towsley, D., Kurose J., "A Comparison of Sender-initiated and Receiver-initiated Reliable Multicast Protocols," SIGMETRICS'94.

[9] Postel, J., "Transmission Control Protocol - Darpa internet Protocol Program Specification," RFC 793, September, 1981.

[10] Saltzer, J.H., Reed, D.P., Clark, D.D., "End-to-End Arguments in System Design," ACM Transactions on Computer Systems, Vol. 2, No. 4, November 1984. pp 277, 288.

[11] Yajnik, M., Kurose, J., Towsley, D., "Packet Loss Correlation in the MBONE Multicast Network: Experimental Measurements and Markov Chain Models," Infocom '96.

[12] Yavatkar, R., Griffioen, J., Sudan, M., "A Reliable Dissemination Protocol for Interactive Collaborative Applications," Multimedia'95.

[13] Zhang, L., Braden, B., Estrin, D., Herzoz, S., Jamin, S., "Resource ReSerVation Protocol (RSVP) Version 1 Functional Specification," RFC in preparation.

[14] Stevens, W., Thomas, M., "draft-stevens-advanced-api-03.txt", work in progress.

[15] Gilligan, R., Nordmark, E., "Transition Mechanisms for IPv6 Hosts and Routers", RFC 1933, April 1996.