Abstract—It is well known that request suppression, local recovery and retransmission scoping are the three crucial elements for scalability and efficiency in multicast loss recovery. None of existing multicast loss recovery schemes can simultaneously have good performance in all the three aspects without introducing significant overhead. The scheme proposed in this paper approaches the multicast loss recovery issue from a new perspective and achieves good performance in all the three aspects with very limited overhead. Our analysis shows that the proposed scheme has a significantly better overall performance as compared to existing schemes.

I. INTRODUCTION

Research on multicast loss recovery focuses on two directions: end-to-end schemes and network-assisted schemes. The schemes in the end-to-end direction, such as [1] [2] [3] [4] [5] [6] [7] [8] [9] [10], generally cannot simultaneously have good performance in request suppression, local recovery and retransmission scoping, which are the three well-known elements for efficiency and scalability in multicast loss recovery. In addition, these schemes either can not adapt to changing network conditions or usually introduce significant control overhead in traffic. The source of problems for end-to-end schemes is the lack of information about the underlying network. Therefore, efforts have also been made to explore schemes in the network-assisted direction, such as [11] [12] [13] [14] [15] [16] [17] [18] [19]. Although these network-assisted schemes, in general, can achieve better overall performance than end-to-end schemes, they either introduce significant overhead (e.g., data caching at router sites) or still have low performance in request suppression, local recovery or retransmission scoping. The scheme proposed in this paper approaches the multicast loss recovery issue from a new perspective and simultaneously achieves good performance in request suppression, local recovery and retransmission scoping with very limited overhead.

It has been shown that distributed multicast loss recovery schemes generally outperform source-based multicast loss recovery schemes in efficiency and recovery latency [20]. However, existing distributed schemes, end-to-end or network-assisted, usually need to prepare a loss-recovery structure before or at the beginning of a multicast session (e.g., distributing distance information among receivers or setting up states in routers across the multicast tree). Even if there is not a single loss during a multicast session, these schemes still set up the structure across the multicast tree. This is not efficient, since setting up a loss-recovery structure requires spreading control messages and sometimes activating states in related routers. To overcome this drawback, the proposed scheme does not prepare a loss recovery structure across a multicast tree in advance. Instead, it only proceeds to set up a minimum loss recovery structure when a loss event does occur. When losses occur on a branch, the scheme agent residing in the router just above the branch dynamically defines a loss region and starts to find a pair of receivers near the loss site to supply loss information and repair packets, respectively. The repair packets are then subcast to the loss region by the scheme agent. After the loss event ends, the scheme agent deletes related states and the pair of receivers release their roles.

Using this new active approach and by involving only local nodes and routers in the recovery process, the proposed scheme can save considerable network resources in the loss recovery for a multicast session. In addition, the proposed scheme simultaneously meets the three well-known requirements for efficiency and scalability in multicast loss recovery. With the proposed scheme, retransmission requests are produced only by the request source instead of all receivers in each loss region. Furthermore, these requests are transmitted to the repair source by unicast instead of multicast or hop-by-hop extra-processing. Therefore, the proposed scheme suppresses requests not only in production but also in spreading. Because repair packets are subcast at the branch where losses occur, the leakage of repair packets to unrelated receivers is impossible. All these measures allow the proposed scheme to achieve better request suppression and retransmission scoping than other schemes that have been proposed (e.g., SRM [5] and LMS [14]). Also, the proposed scheme does not require data caching at router sites to achieve ideal local recovery. Our analysis shows that the proposed scheme has a significantly better overall performance as compared to other existing schemes.

The rest of the paper is organized as follows. The next section presents the proposed scheme. Section III and Section IV show our analysis and simulation results, respectively. Our summary appears in Section V.

II. THE SCHEME

With the proposed scheme, losses at each bottleneck of a multicast tree are recovered right at the loss site. When congestion occurs at a branch of the multicast tree, the router just above the branch sends out control messages to search for a repair source and a request source. After the repair source
and the request source are found, the request source provides loss information to the repair source, while the repair source provides repair packets to the router to subcast. Since each loss is recovered right at the site where it occurs, every receiver finally gets all the data sent from the multicast source, although most receivers do not actively participate in the loss recovery.

Some definitions are given below to assist the detailed description of the loss recovery process:

- A reference branch is a multicast branch that is congested and under loss recovery.
- A reference router is a router that is just above a reference branch. 
- A good branch is a multicast branch that is not experiencing congestion and did not experience congestion recently.
- A stable receiver is a receiver that has been in a multicast group for a period of time (e.g., 30 seconds).
- A direct receiver of a router is a receiver located in the same LAN with the router.
- The NACK_Delay is a parameter whose value decides the delay between a receiver detecting losses and producing a loss report. Initially, it is set to 0.

A. Finding a Request Source

Upon detecting congestion, the reference router sends a message downstream to the reference branch to seek a request source. This message is processed hop by hop. One important field in the header of this message is the ENFORCE, which is used for seeking an optimal request source. An optimal request source is a receiver located nearest to and downstream from the reference branch. Furthermore, there is no other congested link on the path connecting the reference router and the receiver. In this case, the request source only reports losses that occur at the reference branch, so interference is reduced (details about interference appear later). At the beginning of the search process, the message is sent with ENFORCE set to 0. When a router receives this message, it only forwards it to downstream good branches if the router does not have a direct receiver. Otherwise, the message is forwarded to one of the direct receivers only. If no receiver responds in the first round search, the reference router sends a message again but with the ENFORCE set to 1 and this message is repeated until a response is obtained. When a router receives this message, it forwards the message to all downstream branches if the router does not have a direct receiver. Otherwise, the message is forwarded to one of the direct receivers only. If no receiver responds in the first round search, the reference router sends a message again but with the ENFORCE set to 1 and this message is repeated until a response is obtained. When a router receives this message, it forwards the message to all downstream branches if the router does not have a direct receiver. Otherwise, the message is forwarded to the direct receiver only. In this way, the reference router tries to find an optimal request source first; if it cannot find an optimal request source, it seeks a sub-optimal one.

B. Finding a Repair Source

As soon as the request source is found, the reference router sends out a message to search for a repair source (some information from the request source is piggybacked with this message; e.g., some loss information). The search proceeds segment by segment along the path connecting the reference router and the multicast source until a repair source is found. Three fields in the header of the message decide the specific area the current round of search covers. They are the TTL, the Start Hop and the Depth fields (at each hop, TTL is decremented by 1, but the other two fields stay unchanged). TTL and Start Hop decide which segment is searched, while Depth decides how far the search should go downstream along each branch of the segment.

When a router receives the message seeking a repair source, it may take one of the following three actions. First, if the router has a direct receiver, it forwards the message to the direct receiver. Second, if it does not have a direct receiver and the message comes from upstream, the router forwards the message to all downstream good branches. Third, if it does not have a direct receiver and the message comes from downstream, the TTL and the Start Hop fields of the message are checked. There are two possible results for the check. (1) If the TTL is greater than the Start Hop, the message is forwarded upstream only. (2) If the TTL is less than or equal to the Start Hop, besides being forwarded upstream, the message is also copied and forwarded to all downstream good branches except the one where the original message came from. Meanwhile, the TTL of each copied message is set to the value of the Depth field of the original message.

To illustrate the segment-by-segment search introduced above, let us consider the network in Fig. 1. If we want to search for a repair source two hops at a time along the path connecting the reference router and the multicast source, Start Hop should be set to 2. Meanwhile, Depth should be set to a preferred value, e.g., 3. The TTL is then changed in each round of search to search a different segment. In the first round, the TTL is set to 2. In this case, RT3 and RT5 will search their downstream receivers at most 3 hops away; RT2 discards the message because the TTL is 0 when the message reaches it. If no response is received in the first round of search, the reference router RT5 starts the second round of search by
sending a new message with the TTL set to 4. In this case, RT3 and RT5 only forward the message upstream, since when the message reaches them, the TTL is greater than the Start Hop. After the message passes RT3, the TTL becomes less than or equal to the Start Hop, so RT1 and RT2 copy the message and send the copies to downstream good branches except the one where the original message came from. Meanwhile, RT0 discards the message since TTL is 0 when the message reaches it. If there is still no response, the TTL will be set to 6 in the next round of search. This search process continues until a repair source is found.

C. Monitoring Recovery and Filtering Interference

Once the request source and the repair source are selected, the recovery process starts. Loss information is fed to the repair source by the request source continuously, while repair packets are sent to the reference router by the repair source and then subcast at the reference branch. In this process, the request source and the repair source monitor each other for possible failure of either part. If either part fails, the other part notifies the reference router to seek a replacement.

Another important thing during the recovery process is to filter out interference. Because a congested branch affects all receivers downstream from it, it is possible that the request source and the repair source of a bottleneck are also under the influence of other congested branches. In Fig. 2, the repair source and the request source of the bottleneck BN2 (RV6 and RV4, respectively) are under the influence of another bottleneck BN1. Because the repairs from the repair source of BN1 (RV5) reach RV6 and RV4 at different times, special situations may arise that cause duplicate repairs to reach receivers downstream from BN1. For example, if packet A is lost at BN1, a repair, A', will be produced by RV5 and be subcast to all receivers downstream from BN1. It can be assumed that A' reaches RV6 at time t1 and reaches RV4 at time t2. If RV4 produces a loss report between t1 and t2, it reports the loss of A to RV6. Since RV6 has already received A' when it gets the loss report, it sends A' to RT3 to subcast. The consequence is duplicate A's for receivers downstream from BN2.

To avoid this kind of interference, the proposed scheme uses the \(NACK_{Delay}\) parameter to compensate for the time difference between \(t1\) and \(t2\). When a request source receives duplicate repairs caused by interference, it increases its \(NACK_{Delay}\) by a small step. The request source will then wait for a period of time decided by the value of \(NACK_{Delay}\) before producing a loss report after it detects losses. In the last example, if \(NACK_{Delay}\) has been adjusted to a right value by previous duplicate repairs before RV4 detects loss A, then RV4 will try to produce a loss report for packet A after \(t2\), but by that time A' from RV5 would have arrived. Thus no loss report will be produced and the event of duplicate A's is avoided. Although the situation in Fig. 2 is not the only one that can cause duplicate repairs, the cause of such duplicate repairs is the same: the difference between the times at which the repairs from the interference bottleneck reach the repair source and the request source of the bottleneck in question. Therefore, the \(NACK_{Delay}\) parameter works in all other interference situations that cause duplicate repairs.

In summary, with the introduction of the idea of good branch and the \(NACK_{Delay}\) parameter, interference between bottlenecks is solved. On one hand, forwarding control messages only to good branches reduces the chance of possible interference. On the other hand, if interference does occur, the request source filters it out with the assistance of \(NACK_{Delay}\).

D. Completing the Recovery Process

With the proposed scheme, the request source decides when the loss recovery for a loss region is complete. This is because the losses at the reference branch of a loss region are experienced by the request source but not the repair source. The request source ends the recovery process when there is no further loss at the reference branch. Once losses stop, the request source sends a message to the repair source to let it release its role.

III. Analysis

This section analyzes the scheme and compares it with SRM [5] and LMS [14]. SRM, LMS and the proposed scheme have one important common characteristic: enabling receivers as potential repair sources but not having data caching at router sites. Basically, SRM spreads loss information across the multicast tree of a multicast session and any receiver that is capable of repairing the losses is a potential repair source. Meanwhile, SRM uses distance-based random timers to suppress duplicate repair requests and repair packets. Although SRM is robust, it usually introduces significant overhead in loss recovery. LMS achieves good performance in request suppression, local recovery, and retransmission coping by maintaining replier-link related states in each router of the multicast tree for each multicast session. The replier links are used for suppressing the spreading of duplicate repair requests and locating a good repair source. With its permanently maintained states, LMS may have advantages in recovery latency over the proposed scheme when the repair or request source cannot be found locally for a loss event. However, local repair and request sources usually exist in a typical multicast
session because it has been shown that losses mainly occur in local networks in MBone [21]. We analyze the three schemes below in typical situations where local request/repair source exists.

The analysis has the following assumptions: First, 50 packets are lost at one bottleneck in one-time congestion. Second, on average, 5 lost packets are reported in one retransmission request, so there are 50/5=10 independent retransmission requests for the 50 lost packets. Third, message processing and queuing delays are the same for all schemes. Fourth, path delays are counted as number of hops for simplicity. The following performance indices are defined for the analysis:

1) The request suppression index: the total number of hops that all requests traverse for a single loss.
2) The local recovery index: the number of hops between the repair source and the nearest receiver that needs repair packets from the repair source.
3) The retransmission scoping index: the percentage ratio of the number of receivers that do not need repair packets but receive them to the number of receivers that do need the repair packets.
4) The recovery latency index: the total number of hops that the request and the repair for a single loss traverse plus other delays introduced by a specific scheme.

Obviously, a lower value of these indices implies better performance. The final comparison results over binary trees and the example multicast tree in Fig. 3 are shown in Table I. This table shows that the proposed scheme has a significantly better overall performance as compared to the other two schemes.

Because of space limitations, we only demonstrate how to calculate the request suppression index of each scheme over a n-level binary tree (a 3-level binary tree shown in Fig. 4 is used for reference). We assume that the congestion occurs on the last hop of the binary tree (it has been shown that most losses in MBone happen in local networks [21]). Because of the symmetry of the tree, it can be assumed that the losses occur on any last hop of the tree (in the 3-level binary tree, we assume that losses occur on the link between RT3 and RT5, LK3-5). One factor that complicates the calculation of the indexes of LMS is the choice of the replier link of a router. We assume that the two downstream links of a router in a binary tree are equally likely to be chosen as the replier link with LMS.

Now we demonstrate how to calculate the request suppression index of each scheme with SRM, each request is multicast to the whole group and traverses the 2(2^n-1) hops of a n-level binary tree. With k losses reported in each request, the index of SRM is \(\frac{2(2^n-1)}{k}\) (we assumed k = 5 in our case).

With LMS, we first take the 3-level tree as an example (as noted before, assuming losses on LK3-5 and 5 losses reported in each request). In this case, either LK3-6 or LK3-5 can be chosen as the replier link of RT3 with a probability of 0.5. If LK3-6 is chosen as the replier link, the request from RV5 traverses 2 hops to reach RV6, so the request suppression index is 2/5=0.4. However, if LK3-5 is chosen as the replier link, the request from RV5 will go upstream to RT2. Again, with a probability of 0.5, the request will go downstream to reach RV7 or RV8 (i.e., LK2-4 is chosen as the replier link of RT2) or go upstream to reach the multicast source (i.e., LK2-3 is chosen as the replier link of RT2). In the former case, the index is 4/5=0.8, while in the latter case, the index is 3/5=0.6. Therefore, the average index of LMS over the 3-level binary tree is \(\frac{2}{5} \times \frac{1}{2} + \frac{4}{5} \times (\frac{1}{2})^2 + \frac{3}{5} \times (\frac{1}{2})^2 \times \frac{1}{5} = 0.55\) (as analyzed above, RV6, RV7/RV8, and the multicast source have a probability of \(\frac{1}{2}\), \((\frac{1}{2})^2\), and \((\frac{1}{2})^3\), respectively, to become the repair source). We can apply the same logic to a n-level...
binary tree with \( k \) losses reported in each request. In this case, the index becomes a series:

\[
(2 \times \frac{1}{2} + 4 \times (\frac{1}{2})^2 + 6 \times (\frac{1}{2})^3 + \ldots + 2(n - 1)(\frac{1}{2})^{n-1} + n(\frac{1}{2})^{n-1}) \times \frac{1}{k}
\]

\[
= \{4 - (n + 2)(\frac{1}{2})^{n-1}\} \times \frac{1}{k}
\]

With the proposed scheme, each request traverse 2 hops to reach the repair source (in the 3-level tree case, the request from RV5 traverses 2 hops to reach RV6), so the index of the proposed scheme is \( \frac{2}{k} \) if \( k \) losses are reported in each request.

The high efficiency of the proposed scheme comes from its active and on-demand construction and maintenance of minimum loss recovery structures. However, this approach also has some disadvantages. The pair search delay before a recovery process adversely affects the recovery latency of the proposed scheme, so other schemes (e.g., LMS) may have shorter recovery latency than the proposed scheme in some situations where a local request or repair source does not exist. In addition, the timeout method used by the proposed scheme to detect failed search or request/repair sources may also cause higher recovery latency when failure does occur.

Another thing that needs to be mentioned is that generally there are two kinds of loss sources in reliable packet transmission: congestion losses and bit-error losses. Congestion losses come from queue overflows in networks, while bit errors happen during packet processing and transmission. Thanks to the high quality of links in existing networks, bit-error losses are negligibly low compared to congestion losses. The active mechanism introduced in Section II only deals with congestion losses but not bit-error losses. If total reliability is required, receivers send requests to the original multicast source to recover the losses that are not recovered by the active mechanism. This does not significantly affect the total performance of the proposed scheme because bit-error losses are negligibly low compared to congestion losses.

IV. SIMULATION RESULTS

This section uses simulations to further test the performance of the proposed scheme. The simulations were conducted over the example multicast tree shown in Fig. 3. Traffic sources with exponentially distributed idle and burst times are used to cause congestion at two bottlenecks, LK7-10 and LK15-16. The capacity of each bottleneck is 512Kbps, while the capacity of other links is 1024Kbps. The link delay of LK1-2, LK1-7, and LK2-4 is 100ms; the link delay of LK7-10 and LK10-15 is 50ms; LK15-19 has a delay of 20ms. Other links have a delay of 10ms.

A. One-Bottleneck Scenario

In this scenario only the bottleneck LK15-16 is congested. The congestion-causing traffic from RV14 to RV18 starts at the first second and stops at the 10th second. The average rate of the congestion-causing traffic is modified to observe the performance of the proposed scheme under different degrees of congestion. Table II shows the total number of losses, the average recovery latency, and the total number of duplicate repairs observed by RV16 with different congestion-causing traffic rates. As shown in the table, the proposed scheme has low recovery latency. When the average rate of the congestion-causing traffic is 70kb/s, the average loss recovery latency is less than 300ms. As shown by the recovery latency distribution in Fig. 5, most lost packets are recovered in 100 to 200 ms in this case. Also shown in Table II, the average recovery latency is just a little over 600ms even when the losses are pretty heavy.

B. Two-Bottleneck Scenario

In this scenario both bottlenecks, LK7-10 and LK15-16, are congested. To test the proposed scheme with different degrees of interference, simulations were conducted with LK15-16 set to different delays. The simulation results for two cases, 10-second and 20-second congestion duration, are shown in Table III.

In Table III, the average recovery latency and the number of duplicate repairs, in general, go up with the increase of the delay at LK15-16. This can be explained by the following observations. The difference between the times at which the repairs from LK7-10 reach the request source (RV16) and the repair source (RV19) of LK15-16 is partly decided by the delay of LK15-16. Larger difference between the times means heavier interference. With heavier interference, RV16 needs to detect more duplicate repairs before it can adjust its \( NACK_{Delay} \) parameter to a right value. So the number of duplicate repairs may go up with the increase of the delay at LK15-16. Both the increased delay at LK15-16 and the consequently increased value of the \( NACK_{Delay} \) parameter contribute to the increase of the average recovery latency when the delay at LK15-16 is increased. However, there is one exception. When the delay of LK15-16 is 20ms, the recovery latency and the number of duplicate repairs are not significantly greater than that in the 10ms-delay case. This is because the delay at LK15-19 is also 20ms. In this case, the repairs from the interference bottleneck LK7-10 reach RV16 and RV19 almost at the same time if the difference of the queuing delays along the two paths to RV16 and RV19 is not
considered. Therefore, the interference is the least when the delay of LK15-16 is 20ms.

Another thing shown in Table III is that the length of the congestion event does not significantly affect the number of duplicate repairs received by receivers. As shown in the table, there is little or no increase in the number of duplicate repairs when the duration of the congestion event increases from 10 seconds to 20 seconds. In fact, from the introduction of interference in Section II, we know that the number of duplicate repairs produced in a congestion event is mainly decided by the difference between the path delays to the request source and the repair source of the interfered bottleneck from the interference bottleneck, which does not depend on congestion duration.

V. SUMMARY

The scheme proposed in this paper approaches the multicast loss recovery problem from a new perspective: active injection of repair packets. Instead of passively waiting for loss reports from receivers like existing multicast loss recovery schemes, the proposed scheme actively recovers losses right at each loss site upon detecting loss events. When a loss event occurs, the scheme actively sets up a minimum loss recovery structure to recover the losses and maintains the loss recovery structure on-demand. Also because of this new approach, the proposed scheme achieves good performance in request suppression, local recovery and retransmission, and scalable delivery of bulk data. The three well-known elements for efficiency and scalability in multicast loss recovery. All the above features of the proposed scheme contribute to its high efficiency in loss recovery. Our analysis shows that the proposed scheme achieves a considerably better overall performance as compared to other schemes like [5] and [14].

REFERENCES


<table>
<thead>
<tr>
<th>Congestion Source Rate (Kb/s)</th>
<th>Number of Losses</th>
<th>Number of Duplicate Repairs</th>
<th>Average Repair Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>66</td>
<td>0</td>
<td>280.8</td>
</tr>
<tr>
<td>80</td>
<td>112</td>
<td>0</td>
<td>427.3</td>
</tr>
<tr>
<td>90</td>
<td>180</td>
<td>0</td>
<td>592.8</td>
</tr>
<tr>
<td>100</td>
<td>209</td>
<td>0</td>
<td>630.3</td>
</tr>
</tbody>
</table>

TABLE II

SIMULATION RESULTS OF THE ONE-BOTTLENECK SCENARIO

<table>
<thead>
<tr>
<th>Congestion Duration: 10 Seconds</th>
<th>Congestion Duration: 20 Seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay of LK15-16 (ms)</td>
<td>Number of Losses</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>10</td>
<td>120</td>
</tr>
<tr>
<td>20</td>
<td>140</td>
</tr>
<tr>
<td>30</td>
<td>160</td>
</tr>
<tr>
<td>40</td>
<td>180</td>
</tr>
<tr>
<td>50</td>
<td>200</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Congestion Source Rate (Kb/s)</th>
<th>Number of Losses</th>
<th>Number of Duplicate Repairs</th>
<th>Average Repair Latency (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>70</td>
<td>66</td>
<td>0</td>
<td>280.8</td>
</tr>
<tr>
<td>80</td>
<td>112</td>
<td>0</td>
<td>427.3</td>
</tr>
<tr>
<td>90</td>
<td>180</td>
<td>0</td>
<td>592.8</td>
</tr>
<tr>
<td>100</td>
<td>209</td>
<td>0</td>
<td>630.3</td>
</tr>
</tbody>
</table>

TABLE III

SIMULATION RESULTS OF THE TWO-BOTTLENECK SCENARIO