# A Light-Weight Repair Protocol for the Loss-Free Recording of MBone Sessions

Volker Hilt, Martin Mauve, Wolfgang Effelsberg
University of Mannheim
{hilt,mauve,effelsberg}@informatik.uni-mannheim.de

## Abstract

*With the increasing popularity of MBone conferences, the need arose to archive the multimedia streams of these conferences. Like all network applications, a conference recorder is subject to packet loss, which introduces gaps into recorded media streams. Such gaps degrade the playback quality of an archived media stream or even disable the correct playback. In this paper, we present a novel scheme for the loss-free recording of media streams. In contrast to existing systems that require the placement of recording caches within the network, our approach enables an end-to-end system design. We present a detailed protocol analysis that confirms the validity of our approach.*

## 1 Introduction

One of the most popular application scenarios for the Internet Multicast Backbone (MBone) is multipeer conferencing using audio, video and interactive media applications (e.g., shared whiteboards) [7]. The need to archive the multimedia data streams exchanged during such conferences became apparent very early and several tools (e.g., the IMoD [4], MMCR [5], MASH Archive System [8]) were developed that accomplish this task.

Like all networked applications, an MBone recorder is subject to packet loss, which leads to missing data in a recorded archive. In contrast to packet loss encountered by a participant of a live conference, the effects of missing data in an archive are multiplied by the number of times the archive is played back. Data gaps in an audio or video archive degrade the perception quality of the playback of that archive. Gaps in a recorded interactive media stream may entirely disable the proper playback of that archive. For these reasons, it is important to equip an MBone recorder with mechanisms for the loss-free recording of multicast conferences.

In this paper we present a novel approach that enables the loss-free recording of multicast conferences by combining elements of a reliable multicast protocol with caching. In contrast to existing approaches, we do not require the complex placement of heavy-weight recording caches within the network. Our approach is light-weight with regard to resource requirements and it can be implemented completely in the end-systems, following the end-to-end design principle [1] of the Internet. Furthermore, it does not burden the network while the session to be recorded is active.

The remainder of this paper is structured as follows: Section Two describes the relevant research in this area. Section Three describes the Distributed Repair Caches for Multicast Recodings (RCR) that enable the loss-free recording of multicast conferences. Section Four provides a performance analysis of the RCR system and compares our approach to existing work. Section Five concludes the paper with a summary and an outlook.

## 2 Current Research

### 2.1 Synchronous Approach

In the synchronous approach, a traditional reliable multicast protocol is employed to produce loss-free recordings of MBone media streams. Typically, this approach is used with conferencing applications that transmit their media streams over a reliable multicast protocol (e.g., shared whiteboard streams like those of the MBone wb). To record these streams, a recorder must implement the reliable multicast protocol of this application and use the primitives of this protocol to assure the receipt of all relevant data.

The advantages to this approach besides its simplicity are that applications do not have to be modified for recording and that the recordings are available for playback right after the end of a conference. The disadvantage is that additional data is transmitted during the conference to compensate loss encountered by the recorder. These additional packets may interfere with the regular conference traffic and degrade the perception quality of the conference. In

some situations, a repair of losses using a traditional reliable multicast protocol might even not be possible at all, for example if the recorder is connected to the conference via a very congested link.

### 2.2 Asynchronous Approach

To avoid the transmission of additional data within the conference, it has been proposed to store all data sent in recording caches and let the recorder request missing packets after the end of a conference [6][9]. A possible location for such a recording cache is in the end-system of the sender. A drawback to this approach is that it requires a large amount of storage space to be available in the sender. Furthermore, it requires a sender to stay connected to the network until the recorder has requested and received all data needed to repair a recording.

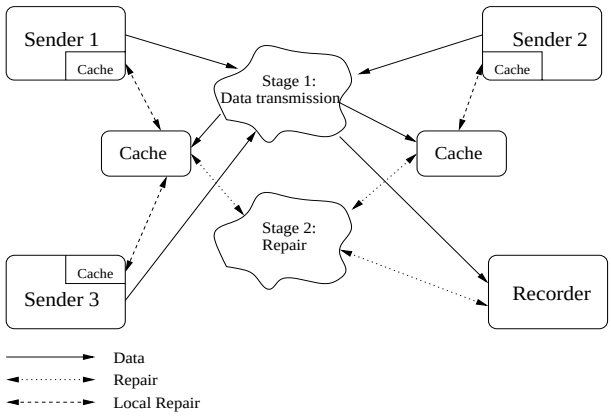


**Figure 1.** Recording caches in network nodes

Another possible location for a recording cache is in the network where a number of caches could be placed such that at least one cache is located near to each sender (see Figure 1). A sender may bind itself to a nearby recording cache, which in turn is responsible for collecting a loss-free copy of data transmitted by this sender [6]. During the recording, the recorder and the distributed caches capture all media data available in the recorded conference. Additionally, a cache assures that all data sent by bound senders is captured without losses, for example by means of synchronous repair mechanisms between cache and sender. At the end of a conference, a recording cache contains the data packets it has received within the conference and all data transmitted by bound senders. After a conference, the recorder initiates the repair of missing data by sending repair requests to the recording caches. These requests may be answered by any cache holding the desired data.

The major advantage to caches placed within the network is that this minimizes the additional tasks a sender has to execute for the recording of a conference. If local repair is not used, a sender does not have to do anything at all during the recording. Thus, any kind of media stream can be recorded without requiring support from the conferencing application. However, if no local repair is used, some data might get lost during the transmission from sender to cache and thus a loss-free recording cannot be guaranteed. If local repair is used, it must be supported by the sender.

The major disadvantage to placing caches within the network is the complexity inherent to this approach. The participants of a typical MBone conference will be distributed over a number of sub-networks, probably owned by different providers. In this scenario, a recorder will not be able to place caches with huge storage requirements within the network near to each sender. Furthermore, it is no trivial task to determine a good placement of recording caches in such a way that one of them is “near” to each sender. If the caches are not placed appropriately, the possibility of losses between sender and cache increases. This results either in a lossy recording or in an increased amount of local repair, which interferes with the regular conference traffic. Even if the caches are properly placed within network, a sender must be able to locate the best cache to bind to. Summing up, recording caches within the network require that the complex task of placing and binding caches with large storage requirements is executed by middleware within the network. Such a system violates the end-to-end design principle of the Internet [1] and is far from being available in the common network infrastructure.

## 3 Distributed Repair Caches

### 3.1 Design Issues

The following design issues should be considered by a system for the loss-free recording of MBone conferences:

- The system should support senders that are distributed over heterogeneous networks - it should not require the presence of a specific infrastructure or middleware in a sub-network. In large-scale conferences with participants connected to a number of different networks, it is unrealistic to assume that a specific infrastructure or middleware is present or can be established everywhere within a short time.
- The quality of a live conference should not be affected by the recording. In particular the network should not be congested by a large amount of additional data for the recording.
- The end-systems should not be overly burdened by helping with the loss free recording (e.g., in terms of storage space or processing power).
- The system should scale well to a large number of par-

ticipants present in the recorded conference. Furthermore, the system should support multiple recorders capturing the same session.

- The system should not be restricted to specific media streams but should support all media types that can be archived by a recorder.
- Although it would be nice to have a loss-free recording available immediately after the end of a conference this is not a requirement of the system. Rather the system should be able to generate a perfect recording within a reasonable amount of time after the conference.

These design issues were considered in the Distributed Repair Caches for Multicast Recodings (RCR). Compared to *recording caches* collecting all the data available in a session, *repair caches* only store data that is actually needed to repair a recording. Thus, repair caches are very light-weight and can easily be placed on the end-system of a sender.

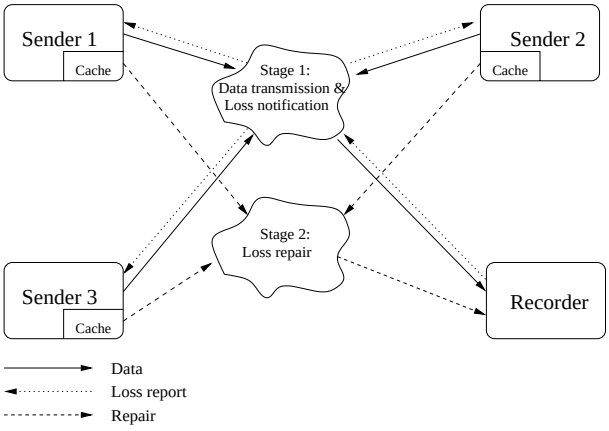


**Figure 2.** Distributed repair caches

## 3.2 System Architecture

The architecture of the RCR system consists of a MBone conference recorder located on an arbitrary host connected to the MBone and light-weight repair caches that are located on the hosts of sending conference participants (see Figure 2). During the recording, the conference recorder captures all data it receives from all senders. The task of the repair caches at each sender is to store exactly the data transmitted that is required to repair losses in the recorded file. The caches capture data only from the local participant. At the end of a recording (or whenever there is enough bandwidth available), these caches transfer the collected repair data to the recorder. Thus, after the repair is completed, the recorder holds a perfect copy of the recorded conference.

In the design of the RCR system we have separated two major tasks: The transmission of loss notifications between recorder and caches, which is executed synchronously during a recording, and the transmission of repair data at a later time. The synchronous transmission of loss reports minimizes the size of the caches since all data can be discarded that will not be needed for the repair. The asynchronous transmission of repair data prevents the conference from being disturbed by the repair data.

## 3.3 Data Transmission and Loss Notification

When the recording of a conference is started, a recorder transmits a description of the session to be recorded and a unique identifier for the current recording to all senders. The session description enables a sender to see if the transmitted media stream is part of the recording. After this initialization, the recorder starts to capture all data packets for the recorded media streams that it is receiving. At the same time, each sender starts to copy all data it is sending into a local cache. This can be done either directly by a recording-aware conferencing application or by a recording agent that receives the data via local multicast loopback from an application. By using such an agent, arbitrary conferencing applications can be recorded loss-free.

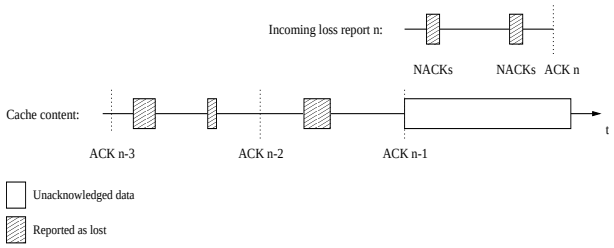


**Figure 3.** Acknowledgment of cache content

At regular intervals, the recorder transmits a loss report to each sender from which the recorder is collecting data. A loss report can either be transmitted point-to-point from the recorder to each sender, or all loss reports can be aggregated into a compound packet and transmitted via multicast to the group of senders. A loss report consists of an acknowledgment (ACK) and a possibly empty list of negative acknowledgments (NACKs). The ACK acknowledges the proper receipt of packets from a sender in the interval since the last ACK sent in the previous loss report (see Figure 3). Each NACK specifies a packet the recorder is missing in the acknowledged interval. On the receipt of a loss report, a sender removes all packets in the acknowledged interval from the cache except of those that are NACKed. The first loss report transmitted to a sender acknowledges exactly the first packet received from that sender. This enables the sender to detect packets missing at the beginning of a recording, which in turn can be stored within the

cache. During the recording, a repair cache holds all packets from the local sender that have not been received properly by the recorder, and all packets that have not yet been acknowledged. At the end of a recording, a cache holds exactly the packets that are required to repair losses in the recording.

Loss reports must be transmitted reliably between recorder and cache. If they are transmitted point-to-point, TCP can be used to assure reliable transmission. Since loss reports are very small, the overhead introduced by TCP is comparably high. Thus, UDP is a good alternative for the transmission of loss reports, even in the point-to-point scenario, and it must be used in the multicast scenario. In these cases, an automatic repeat request (ARQ) mechanism like that used by SRM [2] can be applied to assure reliable transmission. In principle, a cache can detect missing loss reports from gaps in the sequence numbers of received loss report packets. Since loss reports are transmitted at regular intervals by a recorder, a cache may also consider a loss report to be missing if it does not arrive within a reasonable time after the report interval has elapsed. This significantly speeds up the repair of missing loss reports, in particular as the report interval will usually be within the range of minutes. Once a missing loss report has been detected, the cache can request its retransmission.

Missing loss reports do not endanger the repair of a recording since data packets are only removed from the sender's cache if they are acknowledged by a loss report. Thus, a missing loss report increases the size of the cache but never causes data to be lost. The loss-free recording of a conference is also assured if a sender crashes. After the failure of the conferencing system, valid data is no longer transmitted and thus, no data need to be cached. As soon as the conferencing system is up and starts sending again, the cache is also operable again.

When the recording is finished, the recorder sends a stop command to all senders. In turn, they stop copying the data to the cache.

### 3.4 Repair of Losses

In the second stage, the repair data must be transmitted from the repair caches to the recorder. This task can be executed by a separate repair agent on the sending hosts and does not require the conferencing software to run. The repair agent must be a resident service that is activated as soon as data is stored in the cache and that sets itself to passive as soon as all cached data is transmitted.

The transmission of repair data should be started as soon as possible. However, no other activity on the senders host should be disturbed, and the transmitted repair data should not interfere with other network traffic from that host. Since these parameters can best be monitored on the sender host, it should be the sender who decides at which time the transmission of repair data is appropriate. For example, repair data can be sent to the recorder each time the CPU and the network interface are idle. This sender-initiated transmission of repair data copes perfectly with sender hosts that are disconnected from the network after the end of a conference. As soon as the host is connected again, it can resume the transmission of the remaining repair data.

In advance of sending repair data to a recorder, a cache must transmit the unique recording identifier it has received during the initialization of the recording. Otherwise the recorder could not associate the repair data with the right recording. The repair data must be sent reliably to the recorder either using TCP if the session was recorded by only one recorder, or by means of a reliable multicast protocol if multiple recorders benefit from the repair data. Once all repair data has been transmitted successfully, it can be removed from the cache.

## 4 Performance Analysis

### 4.1 System Performance

Due to the end-to-end nature of the RCR system, a performance analysis must consider the costs that are introduced in the end systems involved. In particular, three aspects of the protocol are of interest: the storage space required by the cache during the recording, the bandwidth required for the RCR system during the recording, and the size of the cache remaining at the end of a recording before the repair is done.

The final cache size $S_c$ remaining after a conference can be calculated from the mean loss rate $p$, the mean data rate sent $D$ *(Kbits/s)*, and the duration of the recording $r$ *(seconds)* as follows:

$$S_c = \frac{Drp}{8192} \quad (MB)$$

During the recording of a conference, a certain amount of cache storage is consumed by recent packets that have not yet been acknowledged. This storage space is required in addition to the space consumed by already NACKed packets. It grows dynamically until a loss report is received, which frees some of that space. The average size $S_{cu}$ to which this storage space grows provides a good approximation for the amount of unacknowledged data in a cache for media types with a constant data rate. For applications that typically produce bursty traffic (e.g., shared whiteboards) the amount of storage actually required may deviate from this average value. But even for those media types, $S_{cu}$ still provides a rough estimate of the amount of unacknowledged data in the cache.

In the following calculations we assume that loss reports are transmitted point-to-point via UDP from a recorder to each sender. Furthermore, we assume an uncorrelated packet loss with probability $p$ and an average network delay between cache and recorder of $d$ *(milliseconds)*. The interval $i$ *(seconds)* at which a recorder sends loss reports and the time $j$ *(seconds)* the cache waits before requesting a missing loss report are system parameters that can be set individually for each media type. When a loss report arrives, a cache holds unacknowledged packets that were sent during the last $i + d$ seconds. If the loss report fails to reach a cache, it will be requested $i + j$ seconds after the last loss report was received. If the retransmission succeeds the cache will hold unacknowledged packets that have been sent during the last $i + j + 2d$ seconds. Of course, requests and replies are also subject to packet loss, so that a number of request-reply-cycles might be necessary to receive a loss report.

$$S_{cu} = \frac{D}{8}\left((1-p)\left(i+\frac{2d}{1000}\right)+ p\sum_{z=0}^{\infty}(1-(1-p)^2)^z(1-p)^2\left((z+1)j+i+\frac{2d}{1000}\right)\right)$$

$$= \frac{D}{8}(1-p)\left(i+\frac{2d}{1000}+ p(1-p)\left(\left(j+i+\frac{2d}{1000}\right)\frac{1}{(1-p)^2}+j\frac{1-(1-p)^2}{(1-p)^4}\right)\right)\ (KB)$$

The bandwidth consumed by loss reports depends on the current loss rate in two respects: the size of a loss report increases if more losses need to be reported, and a loss report itself must be requested and retransmitted if it gets lost. A loss report consists of a fixed header part of 46 bytes that consists of IP and UDP headers and the ACK for received packets. For each lost packet, two bytes must be added to signal a NACK. Assuming that the mean size of the data packets sent is $b$ *(bytes)* the average size of a loss report $S_l$ can be calculated as

$$S_l = 46 + 2\frac{1024Di}{8b}p \quad (bytes)$$

If a request for a missing loss report contains 40 bytes (again, including IP and UDP headers), the average bandwidth consumed by the loss reports $B_l$ per sender is

$$B_l = \frac{8}{i}\left((1-p)S_l + p\left(\sum_{z=0}^{\infty}(1-(1-p)^2)^z(1-p)^2((z+1)40+S_l)\right)\right)$$

$$= \frac{8}{i}(1-p)\left(S_l + p(1-p)\left((S_l+40)\frac{1}{(1-p)^2}+40\frac{1-(1-p)^2}{(1-p)^4}\right)\right)\ (bits/s)$$

## 4.2 Example

The following two sample MBone sessions illustrate the bandwidth and storage requirements of the RCR system. In the first example, we consider the recording of a video stream with a data rate of D = 128 Kbits/s per sender and an average packet size of b = 512 bytes. To keep our example simple, we calculate the resource requirements for one sender. We assume a mean loss rate of p = 0.2, which is a high loss rate for the MBone; it can be regarded to be an upper bound up to which the perception of a video stream is not significantly disturbed [3]. Furthermore, we assume an average network delay between sender and recorder of d = 200 ms. The protocol parameter “loss-rate interval” is set to i = 90 s and the “loss report request delay” is set to j = 2 s, which is fast enough for our purposes and makes the protocol robust against network jitter. For a recording of one hour, we get an overall cache size per sender of $S_c$ = 11.25 Mbytes, and the amount of unacknowledged data in a cache grows to $S_{cu}$ = 1454 Kbytes on the average. The mean size of a loss report is $S_l$ = 1198 bytes which enables the unfragmented transmission across most networks. The additional bandwidth required by reports during the conference is $B_l$ = 108 bits/s per sender. Compared to the bandwidth consumed by the video transmission of each sender, the loss reports add an negligible overhead of 0.08%.

In the second example, we take a look at the recording of a shared whiteboard stream. We assume a mean data rate of D = 3 Kbits/second per sender and an average packet size of b = 1024 bytes. Furthermore, we assume a mean loss rate of p = 0.2 and an average delay of d = 200 ms. If the protocol parameters are set to i = 180 and j = 2, a recording of 90 minutes requires an overall cache size of $S_c$ = 405 Kbyte, and on the average, the size of unacknowledged data in the cache grows up to $S_{cu}$ = 68 Kbyte. An additional bandwidth of 4 bits/s will be needed, corresponding to 0.13% of the shared whiteboard traffic.

Summing up, our approach adds a negligible amount of data to be transmitted during a conference, and requires very little hard disk storage in the end systems, making it applicable even to high bandwidth videoconferences in lossy network environments. For low bandwidth media streams, the overhead introduced is minimal.

## 4.3 Repair Caches vs. Recording Caches

The recording caches used the existing approaches and repair caches have different characteristics in a number aspects. In the following section we will compare both approaches.

- **Resource usage** refers to the amount of required storage and bandwidth. Repair caches require that each packet

that a recorder is missing is stored exactly once (by the sender). The storage usage is therefore optimal. We have shown that the required communication overhead to achieve this is negligible. Recording caches store all data that it receives during a session. The resource usage of these approaches is therefore much higher.

- **Robustness** determines how well an approach can tolerate the failure of individual components. Recording caches guarantee a loss free recording only when all caches remain intact. When a single cache crashes all senders that are bound to that cache may be recorded in a less than perfect way. Due to the redundant data in the recording caches it may be possible that a different cache has stored the required repair information. However, this is not guaranteed. In contrast to this, the survival of a sender and its repair cache is related to each other since both are running on the same machine and maybe even share the same process. It is therefore rather unlikely that a sender will be not protected by a local cache. However, when the local repair cache of a sender is wiped (e.g., due to a harddisk crash) then the recorder will be unable to repair the losses from this single sender. If this is not considered sufficiently unlikely then it is possible to increase the redundancy in the repair caches. For example, a sender might buffer the data it has received from certain other senders and which has not been acknowledged by the recorder.
- The **scalability** criterion shows how well an approach can deal with an increasing number of *senders*. For repair caches, the bandwidth required per active sender remains constant and insignificant. The storage requirements remain constant for each sender. In contrast the recording caches require storage that increases with the product of the number of senders and the number of caches. Recording caches are therefore less scalable.
- Finally, the **feasibility** determines how good an approach fits into current system and network environments. The main challenge of the repair cache approach is to keep the burden on the end-systems of the senders at an acceptable level. The analysis in the previous section shows that this is possible. Furthermore it is not unreasonable to expect that the end-system of the sender will, at some time of their choice, be able to transmit the repair information to the recorder in a way that does not disturb the user of that machine. The main problem of the recording cache approaches is the placement of the caches in the network. This requires that Internet service providers would establish recording caches at strategic positions in their network. In addition these caches would have to be quite large. This is certainly not feasible in the near future and might not even be realized in the long run**.**

## 5 Conclusion

We have presented the Distributed Repair Caches for Multicast Recodings (RCR) system that enables the loss-free recording of MBone conferences. The RCR system is based on a novel approach that combines the synchronous transmission of loss reports during a recording with the asynchronous retransmission of repair data at a later repair stage. The synchronous reporting of packet loss keeps repair caches very light-weight and allows their placement directly at the sender. Thus, the RCR system avoids the introduction of complexity into the network and follows the end-to-end design principle of the Internet. An analysis of the RCR system revealed that the storage requirements of our light-weight repair caches are very low and the additional bandwidth required during a conference is usually about 0.1% of the media stream bandwidth, which can be neglected.

**Acknowledgments.** This work is supported by the BMBF (Bundesministerium für Forschung und Technologie) within the “V3D2 Digital Library Initiative II”.

## References

[1] B. Carpenter(Editor). *Architectural Principles of the Internet*, Request for Comments 1958, Network Working Group, IETF, June 1996.

[2] S. Floyd, V. Jacobson, C. Liu, S. McCanne and L. Zhang. *A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing.* In: IEEE/ACM Transactions on Networking, Vol.5, No.6, pp.784-803, 1997.

[3] M. Handley. *An examination of MBone performance*, USC/ISI Technical Report: ISI/RR-97-450, 1997.

[4] V. Hilt, M. Mauve, C. Kuhmünch, W. Effelsberg. *A Generic Scheme for the Recording of Interactive Media Streams.* In: Proc. of IDMS’99, Toulouse, France, pp.291-304, 1999.

[5] L. Lambrinos, P. Kirstein, V. Hardman. *The Multicast Multimedia Conference Recorder*, In: Proc. of ICCCN’98, Lafayette, USA, pp.208-213, 1998.

[6] L. Lambrinos, P. Kirstein, V. Hardman. *Improving the Quality of Recorded MBone Sessions using a Distributed Model.* In: Proc. of IDMS’99, Toulouse, France, pp.377-383, 1999.

[7] M. Mauve, V. Hilt, C. Kuhmünch, W. Effelsberg: RTP/I - Towards a Common Application Level Protocol for Distributed Interactive Media. To appear In: IEEE Transactions on Multimedia, March 2001.

[8] S. McCanne, R. Katz, E. Brewer et. al. *MASH Archive System*. URL: http://mash.CS.Berkeley. edu/mash/overview.html, 1999.

[9] A. Schuett, R. Katz, S. McCanne. *A Distributed Recording System for High Quality MBone Archives*. In: Proc. of NGC ‘99, Pisa, Italy, 1999.