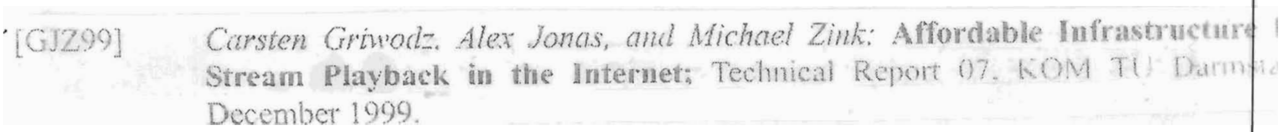
DARMSTADT
UNIVERSITY
OF

Industrial Process and
System Communications
(KOM)

Department of Electrical Engineering
& Information Technology
Merckstraße 25
D-64283 Darmstadt • Germany
Phone: +49 6151 166150
Fax: +49 6151 166152
Email: info@KOM.tu-darmstadt.de
URL: http://www.kom.e-technik.tu-darmstadt.de/



[GJZ99] *Carsten Griwodz, Alex Jonas, and Michael Zink:* **Affordable Infrastructure Stream Playback in the Internet;** Technical Report 07, KOM TU Darmsta December 1999.

Carsten Griwodz, Alex Jonas, Michael Zink

Affordable Infrastructure for Stream Playback in the Internet

Technical Report TR-KOM-1999-07

12. Dezember 1999

Affordable Infrastructure for Stream Playback in the Internet

Carsten Griwodz, Alex Jonas, Michael Zink

KOM - Industrial Process and System Communications
Dept. of Electrical Engineering & Information Technology
Darmstadt University of Technology
Merckstr. 25 • D-64283 Darmstadt • Germany

{Carsten.Griwodz, Alex.Jonas, Michael.Zink}@kom.tu-darmstadt.de

Abstract

The increasing amount of audio-visual (AV) content that is offered by web sites leads to a network bandwidth and storage capacity problem. Caching is one of the techniques that can ease this problem. But even in a caching system the distribution of data (i.e. the AV content) should be bandwidth-efficient in order to preserve the advantages of caching. Furthermore the delivery to the end-user must regard the restrictions implied by real-time data.

This paper describes LC-RTP, an efficient and simple reliable multicast protocol that complies with RTP ([[SCF+96]]). LC-RTP provides lossless transmission of AV content into cache servers and concurrently, real-time delivery to end-users using multicast. It achieves reliability by detecting loss of data and requesting retransmissions after the main session. Therefore the sender transmits the packets with some additional information and listens after the session for any loss requests of the receivers. The retransmission requests list contains ranges of byte counts which can be sent immediately after the main session regarding a sender-defined strategy. This ensures a minimal traffic increase because the transmission of the AV content and any caching will take place while the end-user is served, so dedicated transmissions to the cache are unnecessary. Support for multicast in the distribution system ensures that all cache servers of a multicast group can cache an AV content while transmitting it to a consumer. Whether a server caches a content or not depends on a caching strategy that is chosen independently.

The paper presents a consistent protocol set by combining LC-RTP with the protocols RTSP and SDP that are used for stream control.

Keywords

Video Streaming, RTP Extensions, Caching.

1 Introduction

The increasing interest in transmitting audio-visual data over the Internet shows that streaming is becoming an important application. The huge amount of data in streaming media systems leads to network bandwidth and storage capacity problems. Another problem is the response time, which should be minimal in order to preserve the attractiveness of it. Considering these restrictions and problems it would be advantageous to support such streaming operations with a generic distributed infrastructure, probably implemented by hierarchically arranged proxy caches. A new and popular content could be cached by nodes close to the customer and served to the end-users with low latency, avoiding the use of network resources upstream from the cache server. In this way resources can be used carefully and efficient.

Since network bandwidth is a scarce resource (and we follow the assumption that it will always become scarce again soon after an infrastructure enhancement) and limits the number of concurrent streams

This work is supported in part by a grant of: Volkswagen-Stiftung, D-30519 Hannover, Germany.

(=end-user requests) any unnecessary traffic should be suppressed. Therefore dedicated cache server update transmissions should not be used; rather the caches should receive the content while an end-user located downstream from the cache is served, which is further described as write-through. Furthermore this data transmission could be done via multicast, in order to efficiently update cache servers of a multicast group. Obviously the cache updating mechanism should be reliable in order to have complete and correct copies of the content in each cache server, otherwise errors will be reproduced when the content is streamed from the cache server to the client.

Another problem for the distribution of AV content is the fact that it must be transmitted in real-time. Thus delivery mechanism should care about end-to-end delay, quality of service and the sequence order of the packets. This implies that a client can not wait for any resent packets instead of displaying the current data, so the normal data flow must persist and any retransmission must happen aside of the normal data flow.

A further problem is the resource usage in the cache server itself. The retransmission and storage mechanism should work efficiently without wasting any resources. This means that any extensive buffer usage should be forbidden and the data should be stored efficiently on disks.

This paper describes our protocol set for streaming media delivery. It focuses on LC-RTP, an RFC-compliant extension to RTP for reliable file transfer that requires no infrastructure modifications except on the servers and caches. LC-RTP provides lossless transfer of real-time data by using loss collection (LC). The sender sends RTP-packets via multicast to all receivers (clients and cache servers) in the multicast group. If a cache server detects a packet loss during the transmission it will be memorized in a list. At the end of the streaming session the sender transmits an end packet and all servers that are caching the video from this multicast transmission request the missing parts from the sender. The sender retransmits all missing blocks and waits until no more packets are requested. As the lists are sent with a random delay after the end packet and only one list per server is created, no flooding of the sender will occur.

Based on our own implementation of LC-RTP we did some tests to show that LC-RTP works reliable and perfoms as least as well as tcp-based tranportation protocols.

2 Caching

Caching is a technique that is used in many different ways to improve the efficiency of a system. It is well known and has been used for a long time to improve memory and disk access in computer systems. In recent years a multitude of caching techniques have been used to improve access time in the web (e.g. [[BDH94]], [[Wes96]]).

The performance of a cache depends on the implemented strategies that are responsible for the maintenance of the stored data, as it is typically not affordable to install storage space that can hold all available data. Since most of the existing algorithms for web-caching are designed for optimized access to many small data objects [[Tew98]] it must be examined whether these algorithms are also feasible for AV content. Factors that should be taken into account for streaming media caching are the age of the content, its popularity [[GBW97]], time dependent popularity (hour, day, week) and the size of the file.

A major difference in caching of typical web content up to now and caching for streaming media is the size of data that must be stored and the ratio of storage space to cached items' sizes. On web caches usually large numbers of small files (HTML files of some kbytes) are stored which is different to video caching where one cached item has the size of a least one Gbyte (e.g. 90 minutes of an MPEG-1 movie => 1 Gbyte). Caused by the size of the files that have to be cached, many more files can be stored on a standard web cache than on an AV cache. The large difference between the size of the files also influences the network load and the time that is needed for storing a single item. We have concluded that standard web caching and AV caching have different properties and existing web caching algorithms are not generally applicable to VoD systems.

One possible caching structure which is from our point of view a starting point for further investigations is presented in chapter 3 but we also think that an ideal strategy might have a different layout.

If a client requests an AV content it will be processed by the nearest cache server, leading to a quick response time and less network load in case that the requested content is stored in this cache. In this way further access to the network and other cache servers is not necessary. If the AV content is not present at the first level, than the request is processed by higher levels of the cache hierarchy, consuming more and more resources. An efficient caching algorithm in the distribution system would offer the possibility to increase the overall performance of a system with minimal financial efforts.

3 Protocol Set for Streaming Media

The separation of control and data protocols is a principle approach that has been implemented in Internet video streaming protocols for years, without much consideration about the reasons. Certainly, Distributed Storage Media Command and Control (DSM-CC, [[ISO96]]) is multiplexed in MPEG-2 transport streams, but in on-demand systems this is usually a multiplexing step that is independent from the video stream. The amount of feedback about the stream quality that is transported with the data stream differs from one protocol to another, sometimes stream setup and QoS negotiation are handled in-band with the data stream, but control information such as stream location is exclusively transferred out-of-band.

Recently, the term “HTTP streaming” has been coined. Basically, this is an HTTP GET request for a video file, but the server can draw conclusions about the client actions from the behavior of the TCP stack; this can be considered *implicit signalling* of the control information. It allows the server to determine Start, Pause and Stop actions, and it allows scaling of the content based on the throughput that is experienced at the sender side. The use of TCP makes it non-scalable, but with a different transport protocol, it may be.

We have decided not to work on the latter approach. First of all, the separation of control and data protocols allows the adoption and adaptation of existing protocols. The second reason is that it is also technically favorable because of its modularity. Besides, multiplexing at the network level is always possible, as demonstrated by MPEG-2.

In the Internet, one a set of protocols is currently adopted -partially or completely- by companies in their products for streaming media (Apple, Real Networks, SUN, IBM, Cisco, FVC.com, ...). These protocols are the combination of RTSP/SDP for stream control and RTP/RTCP for streaming. External means are used to locate such content (web pages, email, SAP) and to embed it into multimedia presentations (SMIL, [[Hos98]]).

RTSP/SDP

The Real Time Streaming Protocol (RTSP, [[SRL98]]) is an IETF RFC that is supposed to be used in conjunction with various other protocols. Its functionality is not generic but rather concentrated on stream control. It references elements of HTTP ([[BFF96]]) to which it is weakly related. It can be used with either TCP ([[Pos81]]) or UDP ([[Pos80]]) as an underlying transport protocol. The data transfer protocol that is mentioned in the RFC and that interacts most closely with RTSP, is the Real-Time

Transfer Protocol (RTP, [[SCF+96]]). The same approach applies for the session description protocols; although no fixed session protocol is defined, the RFC specifies the interaction with the Session Description Protocol (SDP, [[HaJa98]]). The protocol is a text-based protocol that refers explicitly to HTTP in parts of its descriptions, and actually it includes several directives from HTTP instead of redefining them. The functionality added in this way includes proxy-support and authentication.

SDP is originally considered as a companion protocol for SAP, the Session Announcement Protocol. However, besides this mode of distribution for session information, others like download from the web or E-mail distribution are also compatible with this kind of information. Basically, SDP provides a line-oriented syntax to describe a multimedia session in ASCII.

RTP/RTCP

RTP (Real-time Transport Protocol) was created to transport real-time data over the Internet. Originally the Internet was created to transport non real-time data belonging to applications like telnet, email, ftp. These applications require correct and complete data transmission without any time restrictions which is given by the TCP/IP protocol. TCP for example has mechanisms to guarantee the correct, complete delivery of data. In contrast to this VoD or other real-time applications make specific time restrictions on how the data is delivered. Internet telephony, MBone-conferences and all video- and audio conferences can not or not satisfactory be realized with the usual protocols. RTP provides functionality to realize real-time applications, but it does not provide any time QoS (Quality of Service) guarantees. QoS guarantees have to be provided through underlying protocols like for example RSVP ([[BZB+97]]). RTP provides payload type identification, sequence numbering, time-stamping, delivery monitoring and supports multicast if the underlying protocol provides this service.

RTP is a protocol independent format to transmit real-time data. Usually it is used over UDP, as UDP allows multiplexing and does not have any retransmission schemes like TCP. A protocol dependent retransmission mechanism would probably violate the time restrictions. RTP is used together with RTCP (RTP Control Protocol) which allows a quality monitoring of the network connection and has minimal control over the session. Furthermore RTCP can be used to identify the sender. The main task of RTCP is to send periodic control packets to all members of the session using the same distribution mechanisms as the data packets.

We have decided to build on these protocols. The resulting protocol set is listed in Table 1, including the tasks that are handled by each protocol.

<table>
<tr><td colspan="2">reliable file transfer & real-time streaming</td></tr>
<tr><td><b>LC-RTP</b>
• RTP-compatible until RTCP BYE message
• use RTP header extensions
• continuous byte count
• retransmission after reception of loss lists</td><td><b>LC-RTCP</b>
• RTCP-compatible user application-defined RTCP packets
• loss-list report receiver -> sender
• retransmission request after random waiting time</td></tr>
<tr><td colspan="2">stream control & sequencing</td></tr>
</table>

RTSP • standard protocol • use SDP	**SDP** • standard protocol • specifies play range • different sources for data segments

Table 1: Protocol set

LC-RTP

RTP with Loss Collections (LC-RTP) implements our idea of a unified protocol for stream transmission that is compatible with RTP, and reliable transfer of content into the cache servers. It solves these problems by making RTP reliable, while the ability is maintained that non LC-RTP capable clients (standard RTP clients) can receive an LC-RTP stream as well.

To describe LC-RTP the transmission process is divided into two parts. The first part works like a regular RTP transmission and ends when end of the content has been transmitted (using the BYE message). The second part follows this BYE message and is used to retransmit all lost data. In this scenario all receivers that are cache servers that have decided to keep the content in the cache, and that have experienced packet loss, will continue to receive packets after the RTP BYE message. Figure 1 gives a general overview of the different steps that are executed during a LC-RTP session.

LC-RTCP

Just as RTP has a companion protocol RTCP for the exchange of information about the data transfer, LC-RTP requires a companion protocol LC-RTCP, which needs to be RTCP-compliant. In application-defined RTCP packets, the receivers inform the sender about their losses after the reception of the BYE packet, unless all of its missing packets have earlier been reported by another receiver.

4 Reliable Multicast

The design of a reliable multicast protocol is determined by the requirements of a specific application or area of applications that the protocol is built for. Real-time applications will accept a lossy data flow but they will not accept a significant delay. This implies that data recovery should not interrupt the flow.

Some examples for reliable multicast protocols are SRM (Scalable Reliable Multicast, [[FJL+97]]), TRM (Transport Protocol for Reliable Multicast, [[SBB+96]]), RMTP (Reliable Multicast Transport Protocol, [[LiPa96]]) and LRMP (Light-weight Reliable Multicast Protocol as an Extension to RTP, [[Lia98]]). TRM and LRMP make similar assumptions about loss detection and repair requests as SRM, so SRM can be discussed as an example for all three protocols. RMTP provides sequenced lossless delivery of bulk data (e.g. Multicast FTP), without regard to any real-time delivery restrictions. It uses a windowed flow control and ACKs for the received packets. This technique allows a reliable transmission, but if packets are lost, the data flow is interrupted because the lost packets are resent immediately by the sender which leads to a non-continuous data stream. So this protocol is not applicable for streaming applications.

SRM is a reliable multicast framework for light-weight sessions and application level framing. It's main objective is to create a reliable multicast framework for various applications with similar needs of the underlying protocol. Each member of a multicast group is responsible for loss detection and repair requests. The repair requests are multicast after waiting a random amount of time, in order to suppress requests from other members sharing that loss. The approach tries to suppress duplicated retransmission requests and duplicated repair packets by randomized timers. As it is possible that the last packet of a session is dropped, every member multicasts a periodic, low rate, session message including the highest sequence number. SRM was tested and implemented in *wb*, a white board application for real-time conferences. It must be mentioned that SRM needs a specific distribution infrastructure which is not widely available in the Internet at the moment.

A third class of reliable multicast protocols are the ones which include FEC (forward error correction) as a technique to achieve reliability [[NBT97]]. Reliable multicast achieved through FEC is also applicable for streaming systems, since usually no retransmissions are necessary during the multicast transmission. The major drawback of this approach is that error correction information appropriate for the client with the worst connection must be included in each multicast packet. This will lead to a higher use of bandwidth thus leading to a reduced connection quality for the clients. In addition a completely new protocol must be built in the case of layered FEC since this model is not compatible with already existing protocols.

With LC-RTP we present a reliable multicast protocol that is applicable for real-time streaming which does not require changes to the infrastructure and which is compatible to standard Internet protocols. It uses an approach that allows a weighted retransmission (sections of the content that are missed by multiple receivers are handled before sections that are reported missing from one receiver only).

5 LC-RTP Design

For usual MBone-conferences with tools like vic ([[McJa95]]) and vat the functionality of RTP is sufficient. As video- or audio streams are transmitted and displayed continuously, small losses within the information are of minor significance. It would be more complicated to retransmit lost data, because they might disturb the normal procedure. With respect to a video-transmission the pictures would be displayed incorrectly and the audio be distorted. But there is a difference in using RTP between cache servers. A cached version of the content in the proxy cache should be stored 100% correctly to avoid error propagation towards the client. With the use of standard RTP, information that gets lost during transmission is also lost to the caches. The problem is that these errors would be transmitted with every stream that is forwarded from the cache server to a client. In any case that should be avoided since it has to be regarded as a degradation of the service quality. The amount of errors would be rising as well in a scenario where content is distributed in a multi-level hierarchy by being stream-transmitted from one cache server to another one located further downstream from the original server. During each transmission data can get lost and thus lead to a higher error rate in stored copies.

LC-RTP solves these problems by making RTP reliable, while the ability is maintained that non LC-RTP capable clients (standard RTP clients) can receive an LC-RTP stream as well.

To describe LC-RTP the transmission process is divided into two parts. The first part works like a regular RTP transmission and ends after the transmission of the original content following by the transmission of a BYE message. The second part follows this BYE message and is used to retransmit all lost data. In this scenario the receiver is a cache server that has received a request from a client but that has recognized that the requested content is not stored locally and therefore a request forwarding to the original or to a cache server located upstream towards the original server is performed. Figure LC-RTP

Communication gives a general overview of the different steps that are executed during a LC-RTP session.

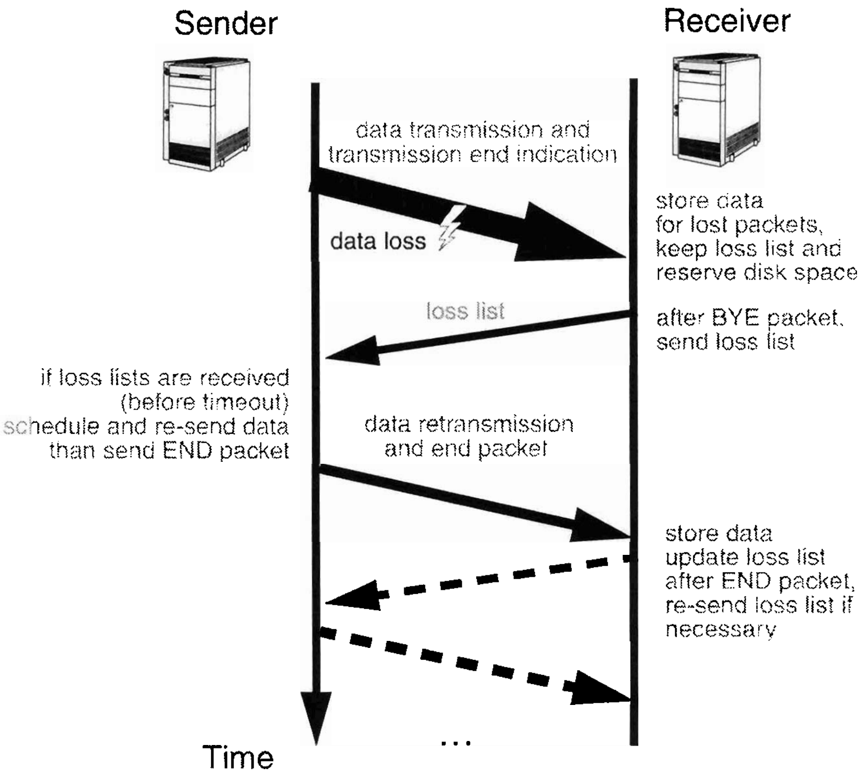

Sender
Receiver
data transmission and transmission end indication
data loss
store data for lost packets, keep loss list and reserve disk space
loss list
after BYE packet, send loss list
if loss lists are received (before timeout) schedule and re-send data than send END packet
data retransmission and end packet
store data update loss list after END packet, re-send loss list if necessary
Time
...

Figure 1: LC-RTP Communication

To explain the functionality the single steps are described in more detail in the following section.

Actions during the content transmission

- SENDER

The sender streams the content that is requested by a client as a multicast stream to all receivers of a multicast group that includes that client. In order to give the receiver the possibility to reserve exactly the required disk space in case of data loss, it is necessary to send information beyond the regular information of an RTP packet. In our case this consists of a byte count. The sender calculates a byte position of the RTP payload, given as the relative position to the stream start, and transmits this information with the data in an extension of the RTP header. A connection between the byte count and the file position of the stored content is not always necessary but can increase cache performance in conjunction with an appropriate buffering strategy or file system.

If possible the byte count should be included in the packet, because it facilitates the synchronization between byte count and the data which are represented by it. For example if the byte count is sent in an extra packet, or via RTCP, the sequence of the byte count and data packet can be changed, or the byte count packet can get lost. If the receiver gets only the data packet, it does neither know whether any data is lost nor how much data is lost. Thus, it is not possible to write the data to the file without buffering large amounts of data or alternatively, without risking time-confusing repair steps in a later repair phase, because there is no information at which position the data should be written in the file.

The byte count can be implemented as an offset-list. By comparing the byte count with the file position of the portion of data that has already been received, exact loss information can be stored in the offset-list. When the sender receives the message of losses, the offset-list can be mapped to the file. If the byte count is equivalent to the number of bytes of RTP payload that has been sent through the network, an encoding-independent storage format can be realized. As a consequence it is possible to have different file layouts on the sender- and receiver side. Each cache server implementation has to transform the mapping of the byte count into its own format. For example one cache server implementation stores the file as raw data and another stores some header information with it.

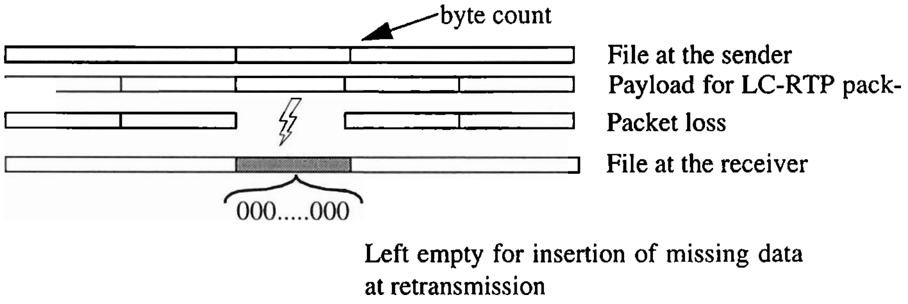


Figure 2: LC-RTP byte count supports retransmission

As a consequence of including the byte count in the data packet, and the requirement of servicing regular RTP clients, only an RFC-conforming protocol extension was an option for us; including the byte count in the payload of the packet would cause problems for standard receivers, like most of the clients are.
At the end of the transmission, an end packet is sent including the last byte count, in order to inform the receivers of the normal end of the transmission including information to check whether data preceding the end packet was lost.

With this end packet the sender has transmitted a whole video as a multicast stream.

RECEIVER

The receiver stores the data and detects a loss by checking the byte count with the last memorized byte count. If a packet loss is detected, the difference between the two byte counts and the length of the actual packet is computed and this computed size can be reserved on the disk for a later insertion of the retransmitted data (see Figure 2). The received payload of the packet is then stored after this reserved gap. Furthermore the loss must be written to a loss list. If no loss is detected the received data is stored on the disk immediately.

Reserving the computed space in the file in case of a loss detection is advantageous be done for several reasons. The first reason is the file system. Most of the existing file systems do not support any efficient insert mechanism, so other mechanisms must be implemented. One conceivable solution would be an index list that contains all the starting points of the packets. With this solution the problem of insertion would be solved, but if a data packet must be searched, a file system seek must performed. As a file system seek consumes plenty of time, it should be avoided. Additionally, either the file system would not behave like a regular file system, or the data would not resemble a regular file.

The solution of reserving the correct amount of space on the hard disk is very simple and efficient, because it preserves the sequential nature of the stored data. And this property is essential for an efficient use of a hard disk, as seeking on a disk importantly diminishes its throughput. Furthermore, this allows LC-RTP to be compatible with multimedia file systems ([[HaSc96]], [[MNR94]]) which are penalized by inserting or do not support it at all.

Actions after the content transmission

- SENDER

After sending the end packet the sender starts a timer. This timer should be a multiple of the worst case RTT (Round Trip Time) between the sender and the known receivers. This RTT can be computed with the periodic RTCP packets that are sent for calculations of the network quality. The relevant value can be a worst case RTT, so no special RTT to a special client or server needs to be stored or computed. During this timer period at least one loss list has to be received from a receiver that has detected packet losses. If the timer runs out without reception of such a loss list, the sender assumes that no loss occurred during the transmission and terminates the session completely.

If a loss list arrives, the requested data is stored in a schedule list. This list includes the requested ranges of data and a counter which indicates how many reporting clients miss this specific data range. The counter is incremented if a loss list from a client arrives that includes a request for data that is already included in the sender's loss list. The counter gives an appropriate strategy some information on a schedule for the retransmission of the lost data. A simple strategy might send the data ranges with the highest loss counter at first, because this ensures that the majority of the cache servers get the lost data early and can then terminate their session and leave the IP multicast group.

Resent packets should be of the same size as the packets that were first sent during the first transmission in order to allow a simple storing mechanism at the receiver's side. The sending mechanism doesn't need to check the range borders but only to check whether the packet has to be stored or not. The byte count that is sent now must be the same as the byte count sent the first time, as otherwise no guarantees of the receiver-sided recognition of the packets can be made. In the same functional procedure as the packet is sent, the schedule list must be updated. This means that the resent data range must be deleted from this list.

When the last entry of the list is processed and deleted, the sender resends the end packet in order to inform the receivers that this retransmission cycle is over. The sender repeats now the procedure of setting a timer and waiting for new possible loss lists to arrive. This procedure is repeated until an application-specific retransmission counter has reached its threshold value or until no more loss lists are sent. The retransmission counter prevents the procedure from repeating endlessly in the case of unexpectedly bad network conditions or in case of misbehaving clients.

RECEIVER

With the reception of the end packet the receiver finishes the normal procedure of the transmission of the content and starts the procedure for initiating retransmissions. To avoid a possible overload of the sender, loss lists are sent from the receivers after a random amount of time. This number should be chosen randomly, but below one measured RTT, where the distance from the sender is considered. The loss list should include all ranges of the detected data losses. If ranges are direct neighbors, they should be combined into one range, in order to keep the size of the list small. This ensures that the additional load of the network remains small. The procedure of sending the loss list after the main transmission ensures that no additional network traffic directed toward the end systems arises during the stream transmission. With this strategy possible network load computations and access control mechanisms need not be changed.

Every retransmitted packet is analyzed to find our whether the byte count in the packet is in the loss list. If it is, the packet is saved at the indicated position in the file by using, if necessary, an offset procedure similar to the one of the sender. Concurrently, the loss list is updated. If the byte count is not included in the loss list the packet is discarded.

When a new end packet arrives, the loss list must be checked. If the list is not empty it has to be sent to the sender again. This procedure is repeated until the loss list is empty, in which case the receiver leaves the multicast group, or until the retransmission counter reaches the application-specific maximum.

To avoid the blocking of a receiver a timer is necessary that terminates the session if no end packet or other resent packets are received after a considerable period.

6 Use and Integration of Protocols

The design of LC-RTP was made within the constraints of an RFC-conforming RTP implementation. Nevertheless the overview gave a general solution of designing a reliable multicast protocols for streaming AV content.

6.1 LC-RTP as an RTP Extension

The main problem in mapping LC-RTP into RTP is the byte count, as it has to be included into the header of RTP (see Section 5). This is necessary in order to keep content of LC-RTP packages compatible with RTP-related packaging RFCs and therefore to make it possible for standard RTP clients to receive LC-RTP streams. Figure RTP header shows an RTP header.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           timestamp                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           synchronization source (SSRC) identifier            |
+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 3: RTP header

The only legal way of inserting the byte count into the RTP header and not into the payload is the use of the extension header of RTP (Figure 4). By setting the x-bit a variable-length header extension to the RTP header is appended. A header extension contains a 16-bit length field that counts the number of 32-bit words in the extension, excluding the four-octet extension header (therefore zero is a valid length).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      defined by profile       |           length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       byte count (64 bit)                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          byte count                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 4: RTP header extension

The other field of the extension header is intended for identifying different header extensions. LC-RTP defines two kinds of header extensions. They are defined to easily distinguish whether a packet is sent as part of the regular stream or during a retransmission phase. The only difference between them is the value in the identifier field. Each extension header has, in addition to the two RTP dependent extension fields, the byte count field. For a current video streaming application this field should be 64 bit long, as a wrap around of the byte count must be prevented. For other applications a simple 32 bit word may be sufficient.

During the usual transmission the RTP transmission is made as usual, except for the byte count which is included in the RTP header. At the end of the transmission an end packet is sent. An appropriate way to do this is by sending an RTCP packet. This packet should not be the normal RTCP BYE packet, as this is used for other meanings. Thus, an application dependent extension RTCP packet must be created. An application defined RTCP packet is shown in figure Application defined RTCP packet.

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|V=2|P| subtype |   PT=APP=204  |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           SSRC/CSRC                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          name (ASCII)                         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   application-dependent data                ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 5: Application defined RTCP packet

LC-RTP defines two application defined RTCP packets. The first one is the end packet and the second one is the loss list packet. The NAME field of both packets is set to LRTP, as it has to be a four digit ASCII name.

The only additional data transmitted in the end packet is the last byte count of the session. The name of the packet itself is of enough information for the receiver to interpret this as the end of the normal transmission. The list appended into the loss list packet should be appended as a list of byte count ranges. If the loss list exceeds the maximal UDP packet size it should be transmitted in several packets. This avoids any congestion problems with the network.

After the loss lists are sent the sender retransmits the lost data by using the extended RTP packets as shown above. These minimal modifications show that the main work of LC-RTP is handled by the logic of the sender and receiver. The extension to RTP is minimal and should be ignored by other applications. In this way LC-RTP is compatible with other applications that participate in the session, like the display tools. This compatibility is very important, because it ensures that a cache server update can be made in parallel to a customer request.

While testing LC-RTP with usual MBone tools an incompatibility was detected. *Vic* and *vat* do not accept any extension to RTP, so they reject all packets with the x-bit set. A comment in the source code explains that an RTP extension is explicitly forbidden through the minimal-control audio and video profile. We have not found any RFC-compliant work-around to this problem, but since *vic* and *vat* implement the variable CSRC list, we have identified at least a non-compliant fix. Since we assume that a cache-based streaming systems would not use mixers, we misuse the CSRC field to transport the byte count instead of the unsupported extension header.

We believe that for the intended application class, the argue that the header extension is sufficiently cheap with an overhead of 8 to 12 bytes per packet. Assuming UDP packets with a typical payload of 512 bytes, our header this causes an overhead of about 1,6%. Furthermore this type of extension is defined in the original RTP RFC ([[SCF+96]]) and should -theoretically- be implemented by all RTP implementations.

6.2 Application-specific SDP Usage

The Session Description Protocol (SDP) has been produced by the MMUSIC working group of the IETF. It was originally intended as a complement for the session announcement protocol SAP to communicate conference addresses and tool-specific information over the MBone. Alternatives such as HTML postings or E-mail distribution of session descriptions were taken into account as well. With this primary goal in mind, SDP does not support negotiation of any of session information, but is just used for dissemination.

With the exception of character encoding rules, this line- and column-oriented protocol is extremely simple. Table 2 shows all of the two character keywords of SDP in the exact order of occurrence in a session description. Keywords must be in first column of a line, without white space before or after the equal sign, and are followed by a set of values on the same line. Carriage return and newline characters determine the end of line, without escaping options.

keyword	meaning	occurrences
v=	protocol version	1
o=	owner/creator and session identifier	1
s=	session name	1
i=	session information	0-1
u=	URI of description	0-1
e=	E-mail address	0-1
p=	phone number	0-1
c=	connection information	0-1
b=	bandwidth information	0-1
time description block		>=1
t=	time the session is active	1
r=	zero or more repeat times	0-1
z=	time zone adjustments	0-1
k=	encryption key	0-1
a=	zero or more session attribute lines	0-1
media description block		>=0

Table 2: SDP protocol format

keyword	meaning	occurrences
m=	media name and transport address	1
i=	media title	0-1
c=	connection information	0-1
b=	bandwidth information	0-1
k=	encryption key	0-1
a=	zero or more media attribute lines	>=0

Table 2: SDP protocol format

We have found SDP appropriate without changes for our purposes. For that reason, this section is restricted to a demonstration of SDP's applicability (in conjunction with RTSP) to the complicated case that the patching/stream tapping mechanism ([[HCS98]], [[GLZ+99]], [[CaLo97]]) is applied transparently to the clients at the caches. Figure 6 gives an example that can be described by the following SDP messages.

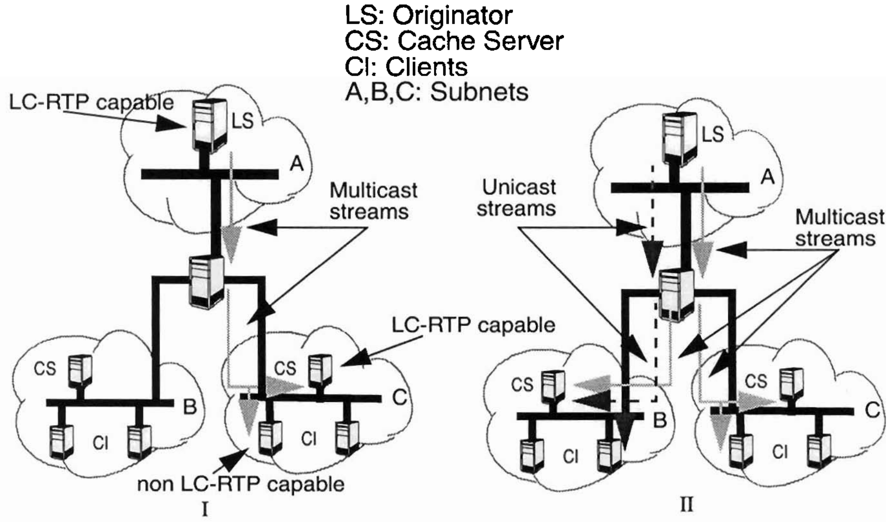


Figure 6: A Possible Caching Procedure

A movie encoded as MPEG system is requested on Oct 17 17:54:46 (3149164486) by an RTP-capable client from its LC-RTP capable proxy cache server (Figure 6 I), and it runs for 90 minutes, i.e. until 19:24:46 (3149169886). This initial viewer will receive the session description of Figure 7. The encoding format is RTP/AVP, which is on the one hand supposed to deceive the client that understands only RTP, on the hand legal, since LC-RTP is RTP compliant. The only deviation from a regular RTP transmission that would be announced by a server is the session attribute *fmtp:lcrtp*, which indicates to the cache servers that our proprietary protocol extension is used as well.

Another user (Figure 6 II) will request the same title five minutes after the start of the movie, i.e. at 17:59:46 (3149164786). When its proxy cache communicates with the original server, it will receive the session description of Figure 8. This session description contains two time fields, the first giving the

```
v=0
o=vsadmin 3149164486 3149164486 IN IP4 192.168.2.1
s=phantclip.mpg
i=The Phantom Menace
c=IN IP4 224.2.24.8/16
t=3149164486 3149169886
k=prompt
a=recvonly
a=fmtp:lcrtp
m=video 49170 RTP/AVP 0
```

Figure 7: SDP specification for an initial LC-RTP stream

original time span, which has already started. The second is the display time of the patch stream, fives minutes from the current time. In the first media description block, information is given that allows to join the multicast stream; in the second media description block, the batch stream is described. It is sent with port information that differs from the original port. This is necessary to allow pass-through delivery of the initial portion of the movie to the client - the packet sequence numbers of the main portion of the movie, which are higher than those that it expects, would force the client to assume major packet losses in its session.

```
v=0
o=vsadmin 3149164486 3149164786 IN IP4 192.168.2.1
s=phantclip.mpg
i=The Phantom Menace
c=IN IP4 224.2.24.8/16
t=3149164486 3149169886
t=3149164786 3149165086
k=prompt
a=recvonly
a=fmtp:lcrtp
m=video 49170 RTP/AVP 0
a=rtpmap:0 MPEG1/1411200
m=video 49172 X-LCRTP/AVP 0
```

Figure 8: SDP specification for an joining LC-RTP streams

In case of support for Patching or one of its variations, it is necessary to support segmented streams and partial retransmission. To support this, another request is re-routed through an LC-RTP-capable proxy server.

The cache server needs to reconstruct the SDP description. Figure 9 shows how the example is modified to include the information that the proxy server is giving to the client to implement a concatenation of the patch stream and the cached stream into a contiguous sequence of a longer one. In this modified SDP description, several details are of interest:

- the *t=* field is now showing start and end times that cover the complete movie length with a time offset appropriate for the 5 minutes that the client has arrived after the original start,
- the *a=fmtp:* line is kept for informative purposes
- the session level line *a=control:rtsp://cache.server.com/phantclip.mpg* indicates that aggregate control is being used; this is necessary and must be enforced by the proxy cache. If the client would be allowed to manipulate the video sessions independently, the situation may arise that the second part of the movie is displayed in parallel with or with an offset from the first part.

```
v=0
o=vsadmin 3149164486 3149164786 IN IP4 192.168.2.1
s=phantclip.mpg
i=The Phantom Menace
c=IN IP4 224.2.24.8/16
t=3149164486 3149170186
k=prompt
a=recvonly
a=fmtp:lcrtp
a=control:rtsp://cache.server.com/phantclip.mpg
m=video 49172 RTP/AVP 0
a=control:patch=1
a=range:npt=0-360
m=video 49172 RTP/AVP 0
a=control:base
a=range:npt=360-324000
```

Figure 9: Pass-through SDP specification moving from the proxy cache to the client

the media level lines *a=control:patch=1* and *a=control:base* are server-chosen names for the stream elements that are delivered.

- the lines *a=range:npt=0-360* and *a=range:npt=360-324000* imply for the client that the second stream needs to be played in sequence with the first one.

7 Conclusions

Performance of LC-RTP

After the implementation of LC-RTP was finished we did some measurement to confirm the assumption that LC-RTP is on one hand reliable and on the other hand performing at least as well as other transport protocols. Therefore we did some long distance measurement between Germany and the USA. A few test have been made so far which indicate that both assumptions are fulfilled by LC-RTP. In the future more performance measurement will be done to consolidate our results.

During the tests we realized that LC-RTP did perform well in point-to-point tests which leads us to the conclusion that LC-RTP must not be used in multicast scenarios only.

Possible Operation Modes

Caching and prefetching of AV content is a powerful method to increase overall performance in the Internet. LC-RTP is designed for this environment. LC-RTP is a simple and efficient reliable multicast protocol compatible with the original RTP. It needs to be implemented only in web servers and proxies. These servers have to be adapted to LC-RTP and they need mainly a list implementation, so the adaptation is a very simple procedure. Other tools are not affected.

All resources are used carefully and the extension permits an implementation to use a simple method to keep the sequential nature of the stored data without buffering. This method considers hard disk performance and possible network structures without wasting resources (like main memory and CPU power). Its intention is to allow a maximum number of concurrent streams handled by the cache servers. As no additional packets are sent during the regular session and the packet sizes are hardly bigger than those of an standard RTP sender, all access control mechanisms and network quality computations can remain unmodified. The only difference to a normal transmission is the fact that after the session, a retransmission of the lost packets to receivers with LC-RTP extensions is performed. A conforming, standard RTP receiver would recognize this as a normal session termination, and would not be affected. Unfortunately, we have observed that popular tools such as *vic* and *vat* do not completely conform to the RTP RFCs. A fix for this situation has been implemented, although LC-RTP's RFC compliance is violated in this case.

By using the same ports as the normal communication, no address conflicts will occur. Multicast ensures a minimum load increase on the network, because the packets are sent only to members of the multicast group, during a transmission to a regular customer.

LC-RTP also supports late joins and early ends of the transmission. The full value of the LC-RTP extension in combination with a special cache server is not yet achieved by simple caching mechanisms. We have already planned a combination of the enhanced Patching technique ([[HCS98]], [[GLZ+99]], [[CaLo97]]) with LC-RTP, supported by RTSP and SDP as shown in Section 6.2, to achieve a relevant decrease in the number of redundant transfers. Since this requires a change in the cache servers' semantics for stream joining (multiple multicast streams must be joint into a single one) we have decided to implement RTP classes with hooks for fine-grained modifications to functional blocks.

8 References

[BDH94] C. Mic Bowman, Peter B. Danzig, Darren R. Hardy, Udi Manber and Michael F. Schwartz, Harvest: A Scalable, Customizable Discovery and Access System, Technical Report CU-CS-732-94, Department of Computer Science, University of Colorado, Boulder, July 1994

[BFF96] T. Berners-Lee, R. Fielding, H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", IETF, RFC 1945, May 1996

[BZB+97] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP), Request for Comments:2205, Network Working Group, 1997

[CaLo97] S. W. Carter, D. Long, "Improving Video-on-Demand Server Efficiency through Stream Tapping", Proc. of ICCCN'97, Las Vegas, NV, USA, September 1997

[FJL+97] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang, A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, ACM Transactions on Networking, 1997

[GBW97] Carsten Griwodz, Michael Bär, Lars C. Wolf, Long-term Movie Popularity Models in Video-on-Demand Systems, Proc. of ACM MM'97, Seattle 1997

[GLZ+99] C. Griwodz, M. Liepert, M. Zink, and R. Steinmetz, Tune to Lambda Patching, In 2nd Workshop on Internet Server Performance (WISP 99), at ACM Sigmetrics '99, May 1999

[HaJa98] M. Handley, V. Jacobson, "SDP: Session Description Protocol", RFC 2327, IETF, April 1998

[HaSc96] R. Haskin and F. Schmuck, The Tiger Shark File System, Proceedings of IEEE 1996 Spring COMPCON, Santa Clara, CA, USA, February 1996

[HCS98] K.A. Hua, Y. Cai, S. Sheu, Patching: A Multicast Technique for True Video-on-Demand Services, Proc. of ACM Multimedia 1998, pp. 191-200, 1998

[Hos98] Philipp Hoschka (Edtr.), "Synchronized Multimedia Integration Language (SMIL) 1.0 Specification", W3C Recommendation 15-June-1998, W3C, June 1998, http://www.w3.org/TR/1998/REC-smil-19980615

[ISO96] ISO/IEC IS 13818, "Information Technology - Generic Coding of Moving Pictures and Associated Audio Information (MPEG2)", ISO/IEC JTC1/SC29, 1996

[Lia98] Tie Liao. Light-weight Reliable Multicast Protocol. Technical Report, INRIA, Le Chesnay Cedex, France, 1998

[LiPa96] John C. Lin, Sanjoy Paul, RMTP: A Reliable Multicast Transport Protocol, INFOCOM 1996

[McJa95] S. McCanne, V. Jacobson, vic: A Flexible Framework Framework for Packet Video. ACM Multimedia'95, November 1995, San Francisco, CA, pp. 511-522.

[MNR94] C. Martin, P. S. Narayan, B. Özden, R. Rastogi and A. Siberschatz, The Fellini Multimedia Storage Server, in Chung: Multimedia Information Storage and Management, Kluwer Academic Publishers, 1994

[NBT97] J.Nonnenmacher, E.Biersack, D. Towsley, Parity-Based Loss Recovery for Reliable Multicast Transmission, ACM SIGCOM 1997, Cannes, France, September 1997

[Pos81] Jon Postel, Transmission Control Protocol (TCP), USC/Information Sciences Institute, Request for Comments: 793, 1981

[Pos80] Jon Postel, User Datagram Protocol (UDP), Request for Comments: 768, 1980

[SBB+96] Bikash Sabata, Michael J. Brown, and Barbara A. Denny, Transport Protocol for Reliable Multicast: TRM, IASTED International Conference on Networks, 1996

[SCF+96] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, Request for Comments: 1889, Network Working Group, 1996

[SRL98] H. Schulzrinne, A. Rao, R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, IETF, April 1998

[Tew98] R. Tewari, Architecture and Algorithms for Scalable Wide-area Information Sytems, Dissertation, University of Texas, Austin, TX, August 1998

[Wes96] D. Wessels, The Squid Internet Object Cache, ICM Workshop on Web Caching,Warsaw, 1996

[WPD88] D. Waitzman, C. Partridg, S. Deering, Distance Vector Multicast Routing Protocol, IETF Network Working Group, November 1988, RFC 1075