# LC-RTP (Loss Collection RTP): Reliability for Video Caching in the Internet

Michael Zink[1], Alex Jonas[1], Carsten Griwodz[1], Ralf Steinmetz[1,2]

[1]KOM - Industrial Process and System Communications
Darmstadt University of Technology
Merckstrasse 25
64283 Darmstadt, Germany
0049-6151-166151

[2]IPSI, German National Research Center for
Information Technology
Dolivostrasse 15
4293 Darmstadt, Germany
0049-6151-869869

*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. 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 ([1]). Its deployment would require neither changes to the network infrastructure nor to existing end-user presentation software. It provides lossless transmission of AV content into cache servers and concurrently, lossy real-time delivery to end-users using multicast. It achieves reliability by retransmission. The traffic increase is minimal because the transmission of the AV content and any caching will take place while the end-user is served. 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. Finally we present the results of long distance file transmissions in order to show that LC-RTP performs well and meets the requirements for lossless transmission of AV content.*

## 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 its attractiveness. Considering these restrictions and problems it would be advantageous to support such streaming operations with a generic distributed infrastructure [2]. A new and popular content can be cached by nodes close to the customer and can be served to the end-users with low latency, avoiding the use of network resources upstream from the cache server.

Since network bandwidth is a scarce resource (and we follow the assumption that it will always become scarce again soon after an infrastructure enhancement) dedicated cache server update transmissions should be avoided. Instead cache servers should receive the content by listening to streams that serve end-users as well. While this can be implemented by multicast, the reliable transfer into the cache must be guaranteed, while the data is also transmitted to the end-user in real-time. The latter implies that an end-user 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.

This paper describes our protocol set that fulfils these combined requirements in the current Internet infrastructure. One of its basic design goal was a protocol set that requires neither costly changes to the network infrastructure nor the replacement of end-user software.

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. It 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 session 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.

Based on our own implementation of LC-RTP we did some tests to show that LC-RTP works reliably and performs as least as well as TCP-based transportation protocols.

## 2. Protocol Set for Streaming Media

In the Internet, one 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.

### 2.1. RTSP/SDP

The Real Time Streaming Protocol (RTSP, [3]) 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 to which it is weakly related. It can be used with either TCP or UDP 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, [1]). 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, [4]).

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.

| reliable file transfer & real-time streaming | |
|---|---|
| **LC-RTP**<br>• RTP-compatible until RTCP BYE message<br>• use RTP header extensions<br>• continuous byte count<br>• retransmission after reception of loss lists | **LC-RTCP**<br>• RTCP-compatible<br>• user application-defined RTCP packets<br>• loss-list report receiver to sender<br>• retransmission request after random waiting time |
| stream control & sequencing | |
| **RTSP**<br>• standard protocol<br>• use SDP | **SDP**<br>• standard protocol<br>• specifies play range<br>• different sources for data segments |

**Table 1: Protocol set**

### 2.2. RTP/RTCP

RTP (Real-time Transport Protocol) was created to transport real-time data over the Internet. VoD, Internet telephony, MBone-conferences and all video- and audio conferences make specific time restrictions on how the data is delivered. RTP provides payload type identification, sequence numbering, time-stamping, delivery monitoring and supports multicast if the underlying protocol provides this service.

Usually it is used over UDP, as UDP allows multiplexing and does not have any retransmission schemes like TCP. RTP is used together with RTCP (RTP Control Protocol [1]) 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.

### 2.3. 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. The functionality of LC-RTP is described in Section 4.

### 2.4. 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 is RTCP-compliant. In application-defined RTCP packets, the receivers inform the sender about their losses after the reception of a BYE packet, unless all of its missing packets have earlier been reported by another receiver.

## 3. 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, [5]), TRM (Transport Protocol for Reliable Multicast, [6]), RMTP (Reliable Multicast Transport Protocol, [7]) and LRMP (Lightweight Reliable Multicast Protocol as an Extension to RTP, [8]). 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 is not applicable for streaming applications,

because the retransmission of the missing data is done immediately after the loss detection.

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. 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. 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 [9]. 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.

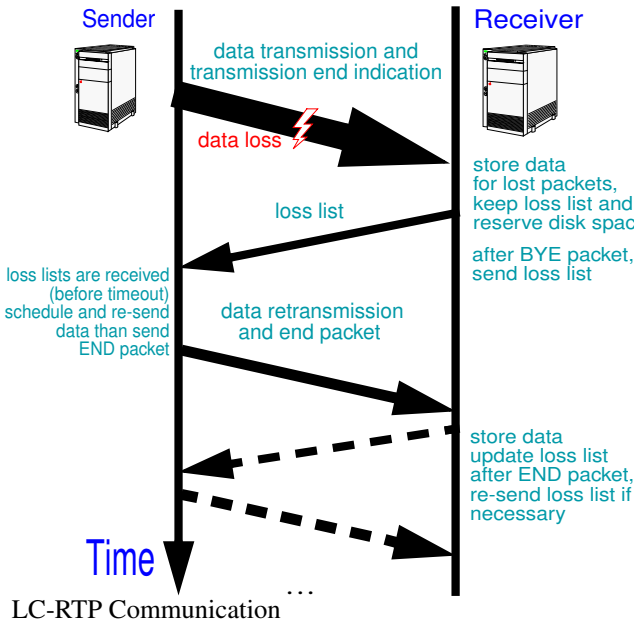
With LC-RTP we present a reliable multicast protocol that is applicable for real-time streaming which does not require changes to the infrastructure and which is compatible to standard Internet protocols. It uses an approach that allows a weighted retransmission (sections of the content that are missed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

## 4. LC-RTP Design

In an environment for AV-caching it is absolutely necessary that the cached version of the content in the proxy cache is 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. 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 1 gives a general overview of the different steps that are executed during a LC-RTP session.



LC-RTP Communication

### 4.1. Actions during the content transmission

The sender streams the content that is requested by a client as a multicast stream to all receivers of a multicast group including 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 which is included in each RTP packet. This mechanism facilitates the synchronization between byte count and the data which are represented by it. If the byte count were sent in an extra packet, e.g. via RTCP, the sequence of the byte count and data packet can be interchanged, or the byte count packet can get lost.

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.

Each cache server implementation has to transform the byte count value into its own file indexing information. As a consequence it is possible to have different file layouts on the sender- and receiver side. For example one cache server implementation stores the file as raw data and another stores some header information with it.

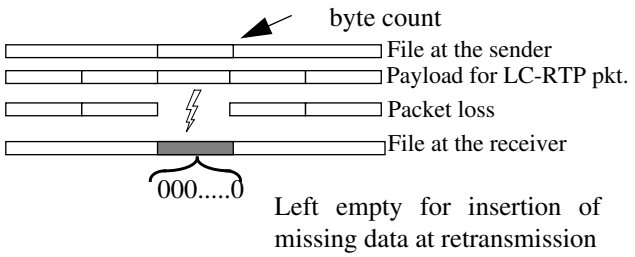



Figure 2: LC-RTP byte count supports retransmission

As a consequence of including the byte count in the data packet, and the requirement of serving 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 (see Section 5.).

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.

Reserving the computed space in the file in case of a loss detection has advantages for several reasons. Our 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 ([10], [11]) which are penalized by inserting or do not support it at all.

### 4.2. Actions after the content transmission

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. During this timer period at least one loss list has to be received from a receiver that has detected packet losses, or the session ends.

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. The loss list includes all ranges of the detected data losses. If ranges are direct neighbors, they are combined into one range, in order to keep the size of the list small.

If a loss list arrives at the server, the requested data ranges are stored in a schedule list. This list includes a counter for each range to indicate the number of requesting clients. This allows the use of a strategy for building a retransmission schedule (e.g. most frequently lost first).

Resent packets are of the same size as the packets that were sent during the first transmission to simplify storing at the receiver. The resent data range is deleted from this list. The client saves each requested, retransmitted packet at the position that is indicated by the byte count. Concurrently, the loss list is updated. If the byte count is not included in the loss list the packet is discarded.

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. This procedure is repeated until an application-specific retransmission counter has reached its threshold value or until no more loss lists are sent.

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.

## 5. Use and Integration of Protocols

The design of LC-RTP was made within the constraints of an RFC-conforming RTP implementation. This section describes how the standard RTP protocol is extended to meet the goal described above.

### 5.1. 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 4.). 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. A 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 3). By setting the x-bit a variable-length header extension to the RTP header is appended. LC-RTP defines two kinds of header extensions. They are defined to easily distinguish whether a packet is sent as part of the regular

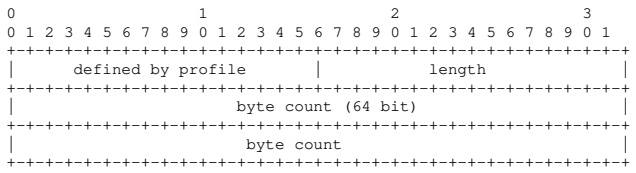



Figure 3: RTP header extension

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 cyclic byte count must be prevented.

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 4.

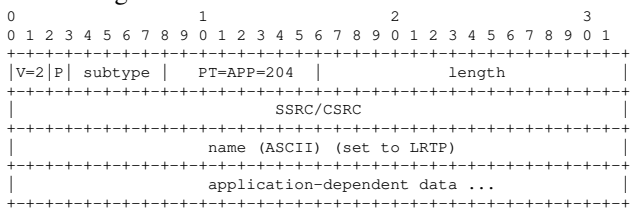



Figure 4: Application defined RTCP packet

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 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.

The extension to RTP is minimal and should be ignored by other applications. This is very important, because it ensures that a cache server update can be made in parallel to a customer request.

During LC-RTP tests we detected that *vic* and *vat* do not accept any extension to RTP, because all packets with the x-bit set are rejected. After examination of the source code we realized that both implementations are not 100% RFC compliant.

We believe that for the intended application class, the header extension is sufficiently cheap with an overhead of 8 to 12 bytes per packet. Furthermore this type of extension is defined in the original RTP RFC ([1]) and should -theoretically- be implemented by all RTP implementations.

# 6. Tests

We finally implemented RTP and LC-RTP in C++.This implementation was used for the tests we performed and which are described in detail in the following.

## 6.1. Test Scenario

Our goal for these tests was to show that LC-RTP performs as well and reliably as other data distribution protocols (e.g. FTP) and can be used for the reliable distribution of AV content.

We transmitted two files (6MB and 20MB of MPEG-I Movie) from locations in Germany, the US and Canada to a receiver located at our institute. We show results from the US (National Institute of Standards and Technology) and Canada (University of Ottawa). The tests were performed 5 times for each file from both locations each time with a different transmission bandwidth. We decided to perform the tests over a larger distance since we expected to have a higher possibility of losses than it might be in a LAN or at connections in Germany.

For each test information about the retransmission was logged at the receiver and the original file and the transmitted file were compared to assure that the transmission completed successful. The comparison for all tests was positive, proving that all transmissions were error-free. We observed that an optimal bandwidth can be found, which results in a mininum transmission time. We observe also that below this optimum, the total number of lost packets per transmission remains the same, i.e. we did not gain reliability from a reduced bandwidth.

| BW [kBit/s] | File Size [MByte] | Max. BW [Bit/s] | | Duration [s] | |
|---|---|---|---|---|---|
| | | NIST | Ottawa | NIST | Ottawa |
| 1000 | 6 | 1047552 | 1022800 | 41 | 42 |
| | 20 | 1024048 | 1024000 | 160 | 160 |
| 2000 | 6 | 2147480 | 2045216 | 20 | 21 |
| | 20 | 2048104 | 2048000 | 80 | 80 |
| 4000 | 6 | 4294968 | 3904512 | 10 | 11 |
| | 20 | 1561080 | 4096000 | 105 | 40 |
| 8000k | 6 | 8593216 | 1169880 | 5 | 37 |
| | 20 | 8192008 | 1058392 | 20 | 151 |
| 12000 | 6 | 8589936 | 1213296 | 5 | 36 |
| | 20 | 5461336 | 487968 | 30 | 337 |

TABLE 2 Test Results (Bandwidth, Duration)

### 6.2. Test results

The results we obtained from the logging we performed during the LC-RTP sessions showed us that retransmissions had to be made in almost all of the tests. The logging information also confirmed that the number of retransmissions increases with the size of the bandwidth we tried to send the files. If the bandwidth is set much higher than the actual bandwidth of the link between sender and receiver multiple retransmissions for one packet are more likely. But also in these cases the files were transmitted without any errors.

During the tests it also became clear that the quality of the link between the US and Darmstadt is of a higher quality than the one between Canada and Darmstadt. We also transmitted both files via FTP from both locations to Darmstadt to obtain some information about the performance of a traditional file transfer protocol.

| File Size [MByte] | Max. BW [Bit/s] | | Duration [s] | |
|---|---|---|---|---|
| | NIST | Ottawa | NIST | Ottawa |
| 6 | 576000 | 328000 | 71 | 126 |
| 20 | 568000 | 304000 | 273 | 512 |

TABLE 3 Test Results FTP

## 7. Conclusions

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, which is stated by the tests we performed. It needs to be implemented only in servers and caches, other tools are not affected. 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.

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.

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 ([12], [13], [14]) with LC-RTP to achieve a relevant decrease in the number of redundant transfers. Loss patterns indicate that the technique would work well as a complement of the FEC [9] proposed by Biersack et. al.

## References


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

[2] C. Griwodz, M. Zink, M. Liepert, G.On, R. Steinmetz, *Multicast for Savings in Cache-based Video Distribution*, to appear in MMCN 2000, San Jose, January 2000.

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

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

[5] 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.

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

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

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

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

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

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

[12] 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.

[13] 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.

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