Darmstadt University of Technology

Exploiting the Fair Share to Smoothly Transport Layered Encoded Video into Proxy Caches

Michael Zink, Carsten Griwodz, Jens Schmitt, Ralf Steinmetz

30 May 2001

Technical Report TR-KOM-2001-10

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/

Exploiting the Fair Share to Smoothly Transport Layered Encoded Video into Proxy Caches

Michael Zink[a], Carsten Griwodz[b], Jens Schmitt[a], Ralf Steinmetz[a]

[a]*KOM, Darmstadt University of Technology, Merckstrasse 25, 64283 Darmstadt, Germany*

[b]*IFI, University of Oslo, Gaustadalléen 23, N-0371 Oslo, Norway*

ABSTRACT

In contrast to classical assumptions in Video on Demand (VoD) research, the main requirements for VoD in the Internet are adaptiveness, support of heterogeneity, and last not least high scalability. Hierarchically layered video encoding is particularly well suited to deal with adaptiveness and heterogeneity support for video streaming. A distributed caching architecture is key to a scalable VoD solution in the Internet. Thus, the combination of caching and layered video streaming is promising for an Internet VoD system, yet, requires thoughts about some new issues and challenges, e.g., how to keep layered transmissions TCP-friendly. In this paper, we investigate one particular of these issues: how can a TCP-friendly transmission exploit its fair share of network resources taking into account that the constrained granularity of layer encoded video inhibits an exact adaptation to actual transmission rates. We present a new technique that makes use of retransmissions of missing segments for a cached layered video to *claim* the *fair share* within a TCP-friendly session. Based on simulative experiments the potential and applicability of the technique, which we also call *fair share claiming* is shown. Moreover, a design for the integration of fair share claiming in streaming applications which are supported by caching is devised.

Keywords: Video on Demand, Wide-Area Caching Architecture, TCP-friendly, Layer Encoded Video

1 INTRODUCTION

1.1 Motivation

In the last few years, the Internet has experienced an increasing amount of traffic stemming from the use of multimedia applications which use audio and video streaming [11]. This increase will continue and even be reinforced since access technologies like ADSL and cable modems enable residential users to receive high bandwidth multimedia streams. One specific application which will be enabled by future access technologies is Video on Demand (VoD). VoD allows clients to watch a large variety of video content via the Internet, from small video clips up to movies. One type of VoD is True VoD (TVoD) [10] which allows users to watch a certain video at any desired point in time and, in addition, offers the same functionality as a standard VCR (fast forward, rewind, pause, stop). The challenges of providing TVoD in the Internet are manifold and require the orchestration of different technologies. Some of these technologies like video encoding are fairly well understood and established. Other technologies like the distribution and caching of video content and the adaptation of streaming mechanisms to the current network situation and user preferences are still under investigation.

Existing work on TVoD has shown caches to be extremely important with respect to *scalability*, from network as well as from video servers' perspective [5]. Scalability, of course, is a premier issue if a TVoD system is considered to be used in the global *Internet*. Yet, simply reusing concepts from normal Internet Web caching is not sufficient to suit the special needs of video content since, e.g., popularity life cycles can be very different [6].

Besides scalability, it is very important for an *Internet* TVoD system to take into account the "social" rules implied by TCP's cooperative resource management model, i.e., to be *adaptive* in the face of an (incipient) network congestion. Therefore, the streaming mechanisms of an Internet TVoD system need to incorporate end-to-end congestion control to prevent unfairness against TCP-based traffic and increase the overall utilization of the network. Note that traditionally video streaming mechanisms rely on open-loop control mechanisms, i.e., on explicit reservation and allocation of resources. As it is debatable whether such mechanisms will ever be used in the global Internet, e.g., in the form of RSVP/IntServ [1], we do not assume these but build upon the current best-effort service model of the Internet which is based on closed-loop control exerted by TCP-like congestion control. Yet, since video transmissions need to be paced at their "natural" rate, adaptiveness can only be integrated into streaming mechanisms in the form of quality degradation and not by "shifting" traffic in the time domain as for elastic traffic like, e.g., FTP transfers. An elegant way of introducing adaptiveness into streaming is to use layered video encodings [9] as it allows to drop segments (the transfer units) of the video in a controlled way without the high computational effort of, e.g., adaptive encodings as described in [21].

However, while the combination of caching and adaptive streaming promises a scalable and "Internet-conform" TVoD system it also creates new challenges for the design of such a system. One particular issue is that video content can only be cached in the form as it has been transmitted, i.e., it potentially consists of successive "steps" of different quality levels corresponding to the different layers. For subsequent requests for that video a decision must be made whether segments from missing layers are retransmitted and if so, which ones, or if some of the cached segments are discarded. The scheduling of these retransmissions affects the perceived quality of the cached video content in a significant way since it is very important that quality variations are minimized as they are disturbing for users [4]. In preceding work [25] we developed and compared different retransmission scheduling algorithms that meet users' demands to watch high quality video with relatively little quality variations. In this paper, we focus on how to perform these scheduled retransmissions in combination with a TCP-friendly transmission method by claiming the fair share for the TCP-friendly session. Transmitting a layer encoded video in a TCP-friendly manner would not always result in the case that the session claims its fair share of network resources. We propose a mechanism, called fair share claiming (FSC), which combines the transmission of a layer encoded video and some additional data, resulting in the utilization of the fair share a session is entitled to.

1.2 Outline

After this motivation, we briefly want to introduce the basic components of our overall approach towards scalable adaptive video streaming in the Internet. In Section 3, we present the initial idea of fair share claiming (FSC) that makes use of the additional bandwidth that is not claimed by the layer encoded video without breaking the cooperative rules implied by TCP's resource allocation model. We demonstrate the applicability of this approach by simulations in a 3-step based simulation environment. In Section 4, we demonstrate how FSC can be integrated into a streaming application by using and partly extending already existing protocols. In Section 6, we summarize our findings, draw some conclusions and give a brief outlook to future work.

2 SCALABLE ADAPTIVE STREAMING (SAS)

2.1 Scalability - Caching

As with traditional web caches, caches for TVoD systems allow to store content closer to users, reduce latency, as well as server and network load and increase the system's fault tolerance. Yet, in contrast to web caches the characteristics of the data to be stored are very different. High quality video files are much larger than most web pages and therefore different caching strategies are used in caches for VoD systems. Furthermore, the distribution process for video files is complicated by the fact that the transmission is much more time and bandwidth consuming. Thus store-and-forward approaches can not be applied.

Let us briefly describe our video caching architecture. As caching method we employ so-called write-through caching*. With write-through caching a requested stream is forwarded "through" the proxy cache to the clients and the proxy cache stores the stream on its local cache if a positive decision is made by the cache replacement strategy. Subsequent clients can then be served from the proxy cache (see Figure 1). This technique reduces the overall network load in a TVoD system compared to a method where the video is transported to the cache in a separate stream using a reliable transmission protocol (e.g., TCP) [3]. On the other hand, write-through caching requires a reliable transport protocol to recover from packet losses. In [23], we present the design and implementation of such a protocol, called *Loss Collection RTP* (LC-RTP), which fits particularly well in a TVoD architecture. It provides lossless transmission of AV content into cache servers and concurrently, lossy real-time delivery to end-users. 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.

2.2 Adaptiveness - Layered Video

Enabling congestion control for streaming applications requires quality adaptation in contrast to elastic applications that allow a reduced transmission speed. However, this quality adaptation does not solely serve congestion control purposes but also satisfies the needs of the large variety of heterogeneous clients that exist in the Internet. Layered video, i.e., video that is encoded in base and enhancement layers which have hierarchical relationships, represents a suitable method to allow for this quality adaptation although there are other alternatives like adaptive encoding or switching between different encodings. Yet, the latter are less attractive for caching purposes since they do not possess the subset relationship of layered encoding and thus

*. Adopted terminology from memory hierarchies.

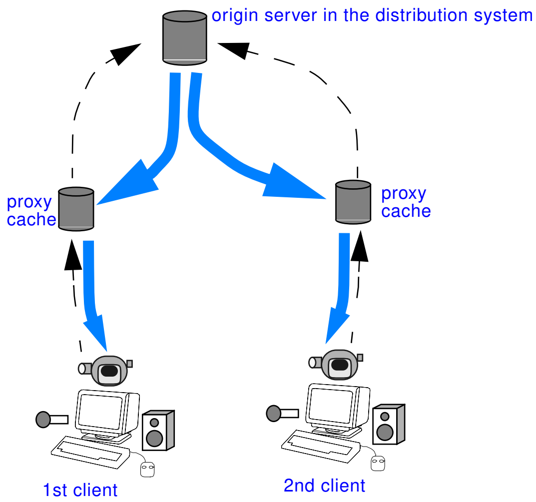

Figure 1: Video distribution.

lead to transmissions which are difficult to cache. Figure 2 illustrates how a layered video might be stored on a cache after its initial (congestion controlled) transmission.

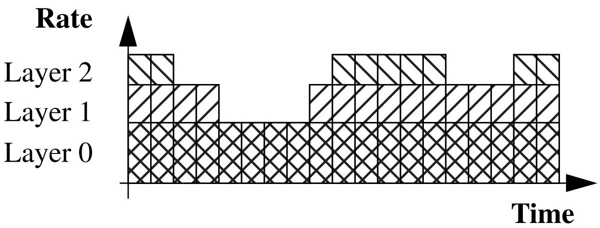

Figure 2: Initial cached video quality.

Obviously, the cached copy of the video exhibits a potentially large number of missing segments from different layers. The exact "shape" of a cached video content is a function of the congestion control mechanism being used. There have been several proposals how to achieve TCP-friendly congestion control using layered video transmissions, e.g., [16] or [2]. Our focus here is on how retransmission scheduling can be incorporated in TCP-friendly layer encoded video transmissions by claiming the fair share for the TCP-friendly session.

3 FAIR SHARE CLAIMING (FSC)

In this section we will present our idea of the fair share claiming mechanism that makes use of the additional bandwidth that is not claimed by the layer encoded video without breaking the cooperative rules implied by TCP's resource allocation model. After a detailed description of the FSC mechanism in Section 3.1 the simulation environment for FSC is presented in Section 3.2 and finally the simulation results are shown in Section 3.3.

3.1 TCP-friendly Streaming

It is not our purpose to develop new TCP-friendly mechanisms for streaming. In recent years several protocols for the transport of non-TCP traffic with TCP-friendly congestion control were developed. Widmer et al. have published an overview of the approaches [22]. To be applicable for streaming over the Internet these protocols have to meet the following requirements:

- Rate oscillations should be kept to a minimum
- Modification to the network infrastructure should be prevented (e.g. the protocol stack in the routers can stay as it is).

From our observations and the classification presented by Widmer et al. [22], *TCP-friendly Rate Control* (TFRC) is very promising as a TCP-friendly protocol for unicast streaming [2]. It is a rate based congestion control protocol with good TCP-friendliness. The main advantage in combination with A/V streaming is that the rate is smooth in the steady-state case and therefore applications that rely on a fairly constant sending rate are supported. In addition, the protocol is end-to-end which does not require any modifications to the network infrastructure. The major drawback of TFRC is its lack of multicast support. The sending rate in TFRC is adjusted to a TCP equation that is based on the loss rate which is transmitted from the receiver to the sender once per round-trip time. In addition it uses slow-start at start-up and for loss recovery.

Transmitting a layer encoded video with the maximum rate that TFRC would allow does not always make sense. If, e.g., the possible transmission rate would be much higher than the actual rate needed for the video, the receiver might need a large buffer to store segments until their playout time is reached. Rate changes in TFRC will not always result in a rate change for the layer encoded video because changes might occur too often and the encoding format will only provide a certain amount of different layers resulting in a finite amount of possible transmission rates. This can result in a situation where the actual possible transmission rate (determined by the TFRC algorithm) and the rate constituted by the sum of several layers might differ. We choose the following example to illustrate the problem in more detail: Let us assume a layer encoded video that consists of up to three layers, each requiring a constant transmission rate of 0.5 Mbit/s that should be transmitted in a TCP-friendly manner via TFRC. At a certain point in time during the transmission the TFRC algorithm determines a maximum possible transmission rate of 1.3 MBit/s. This would allow a transmission of two layers of the layer encoded video whereas 0.3 MBit/s would be wasted if the video would not be transmitted faster than necessary and an additional third layer can not be transmitted because this would require a TFRC rate of at least 1.5 MBit/s. Using the additional bandwidth is the fair share that may be claimed by a corresponding TCP session, yet due to the inelastic and discrete nature of layer encoded video it cannot be claimed. Nevertheless if we find some data to fill this gap we could claim our fair share without breaking the cooperative rules implied by TCP's resource allocation model. Figure 3 depicts an example TCP-friendly layered video transmission. The creation of the TFRC trace is explained in more detail in Section 3.2. In this example scenario the transmission rate for the layer encoded video is only increased in case that the rate determined by the TFRC mechanism would allow the transmission of an additional layer. If this is not the case the additional bandwidth (marked hatched in Figure 3) could be used for the transmission of additional data. In the example shown in Figure 3 that would be an additional 142 MByte. This could be for example data like, e.g., management or statistical information that should be sent from the origin server to the proxy cache. But in this paper, the focus is especially on the retransmission of missing segments of the video that is currently streamed or videos that are already (but not completely) stored on the proxy cache to claim the fair share for this TCP-friendly session. This techniques are referred to as *in-band* FSC for the former case and *out-of-band* FSC for the latter. In [25], we already devised and analyzed different retransmission scheduling algorithms that could be used to determine which of the missing segments should be transmitted. The following simulation shall shed some light on whether the combination of both techniques (retransmission scheduling and FSC) is an appropriate method to improve the quality of a cached layer encoded video. In the following we will present a simulation model to investigate the presented problem in more detail.

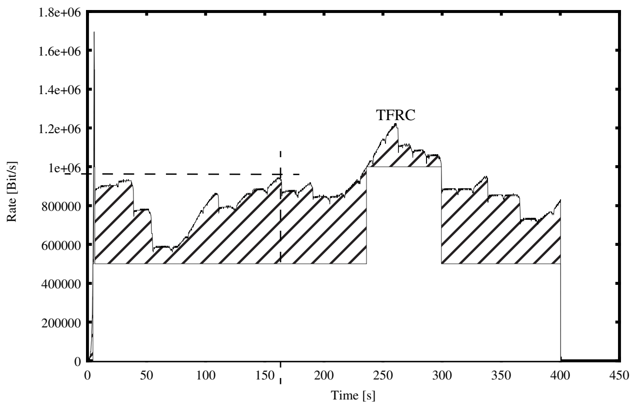

Figure 3: Example layer encoded video transmission via TFRC

3.2 Simulation

We split the simulation in three steps to create a scenario that represents the mechanisms presented in Section 3.1 and to keep the simulation environment more generic:

- Creation of a TFRC trace.
- A possible layer encoded video transmission that is derived from the TFRC trace.
- Determination of segments that can be retransmitted due to spare bandwidth.

The fact that the creation of the TCP-friendly protocol traces is separated from the other two parts allows an exchange of the TCP-friendly protocol without modification of the other parts. Each single part is explained in more detail in the following sections.

3.2.1 Creation of TFRC traces

We decided to perform investigations using ns-2† because a TFRC model is already included and ns-2 allows us to create traces that can be used as a basis for the second simulation step. With the simulation architecture shown in Figure 4 we model a scenario for the distribution techniques we describe in this paper. The simulation architecture consists of 2 routers, 2 senders and 2 receivers. The routers R1 and R2 are connected via a duplex link (L1) that has a bandwidth capacity of 15 Mbps and a delay of 100 ms. This represents a scenario that consists of an origin server S1 and a proxy cache PC1 that caches video streams and forwards them to requesting clients. To model competing web-like traffic between S2 and E1 we use ON/OFF sources as proposed by Floyd et al. [2]. In addition there is a TCP session between S2 and E1 which serves as a reference in order to observe the TCP-friendliness of TFRC. It is active throughout the entire simulation. The ON/OFF sources are also enabled during the whole simulation. One long-lasting TFRC stream, representing the layer encoded video transmission, is initiated at simulation start. A single simulation lasts for 400 seconds. The trace shown in Figure 3 was generated with the method described here.

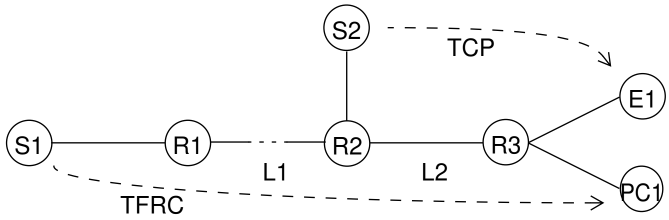

Figure 4: Simulation architecture

3.2.2 Layer encoded video transmission

In the simulations, we presume layer encoded videos that can consist of up to three layers. Here, we assume without loss of generality that all layers are of equal size, CBR encoded and therefore require an identical transmission rate, which is 0.5 MBit/s for these simulations. To create a layer encoded video transmission a TFRC trace created by the methods described in Section 3.2.1 is used as the starting point. We implemented a small C++ program that scans the bandwidth for each entry of the TRFC trace and determines the amount of layers that can be transmitted based on the TFRC bandwidth. The rate for the layer encoded video transmission in Figure 3 was generated in this way. It must be stated here that the strategy for increasing or decreasing one of the layers is very simple and should be subject to further investigations. E.g., a more intelligent strategy might also contribute to a smoother transmission of the video. During the execution of this program an additional list is built that stores information about the spare bandwidth that is available for the transmission of additional data. For the example shown in Figure 3 at 163.2 seconds a spare bandwidth of 452000 Bit/s would be determined. The simulation is discrete since the TFRC trace has a resolution of 0.2 seconds. This restriction had to be made to keep the overall simulation effort in reasonable limits. The error that is introduced by this simplification is negligible since the delay for the link between R1 and R2 is 100 ms and therefore the RTT is at least 200 ms thus leading to the fact that two consecutive rate changes will never be less than 200 ms apart.

3.2.3 Retransmission

To investigate retransmission scheduling in layered video caches in more detailed we already built a simulation environment that is described in more detail in [25]. In contrast to the simulations presented here an instance of layer encoded video was created randomly. The available bandwidth for retransmissions was constant for a single simulation and was only modified to compare the behavior of the retransmission scheduling algorithms in relation to different amounts of available bandwidth. For the simulations in this paper the bandwidth for retransmissions can change in each step of the simulation. For this reason the simulation environment had to be changed in two respects:

- For each step in the simulation the available bandwidth for the retransmission of missing segments must be calculated. This is performed with the aid of the list generated by the simulation tool described in Section 3.2.2 that contains information about the available bandwidth for retransmissions at a certain point in time.

†. http://www.isi.edu/nsnam/ns/

- If retransmissions are performed for the simultaneously streamed video the retransmission scheduling algorithm can only regard the already transmitted part of the video. This is in contrast to our earlier work where we assumed that the complete instance of a cached layer encoded video is known.

In Figure 5, a comprising overview of the three simulation steps is given.

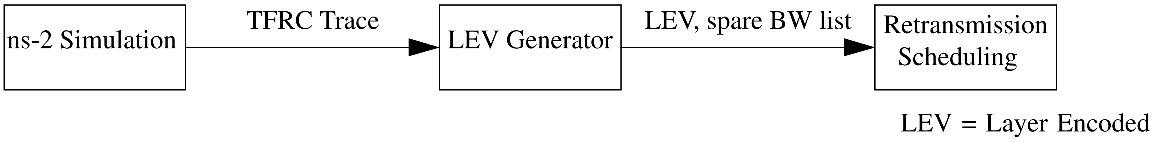

Figure 5: Simulation sequence

3.3 Simulative Experiments

We generated two different kinds of simulations, one for in-band and one for out-of-band FSC. The latter were performed to compare our algorithm [25] against the algorithm presented in [17] (For easier identification of both algorithm we refer to them as *unrestricted* and *restricted,* respectively). Since the latter looks always a certain amount of time ahead of the current playout to determine segments for retransmission it is not applicable for in-band FSC.

3.3.1 In-band FSC

The result of the in-band FSC simulation is depicted in Figure 6. It shows how the quality of a layer encoded video on a proxy cache can be improved with the aid of the FSC technique. In this specific scenario it was possible to add an additional layer for more than half of the length of the complete video. Thus the next client that requests this video from the proxy cache will have the chance to receive it in a significantly better quality than the first client. Unfortunately, there is a small gap for layer 2 between the 200th and 250th second that increases the quality of the cached content. One possible solution to close this gap would be the usage of the out-of-band FSC technique (during the transmission of some other video). To reduce the amount of layer changes the caching strategy on the proxy cache might decide to delete the short amount of the third layer that was cached due to the peak of the TFRC around 5 seconds after the transmission started.

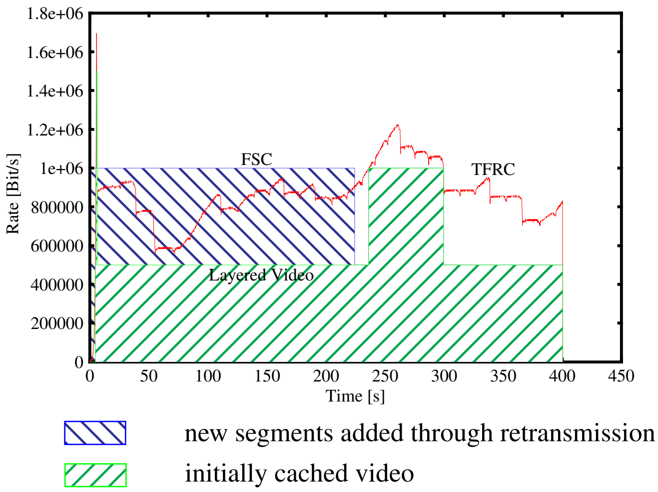

Figure 6: Result of an in-band FSC simulation

3.3.2 Out-of-band FSC

The out-of-band FSC simulation was performed in a slightly different way than the in-band simulation. This is due to the fact that segments for an already cached video are retransmitted. To be able to compare the results of this simulation with the results of the in-band simulation we assumed that the cached video has the same layout as the layered video trace in Figure 6. I.e., we assume the same initial transmission as in the in-band simulation but without any retransmission for this specific video. This will allow us to compare the quality improvement between the in-band and out-of-band technique in the unrestricted case. Since the "layout" of the video is now completely known both algorithms restricted and unrestricted can be applied. We generated a second TFRC trace which determines how much additional bandwidth is available for the retransmission of missing segments. The result of this simulation which is depicted in Figure 7 clearly shows the disadvantages of the restricted algorithm. Caused by the fact that only missing segments ahead of time from the actual playout point are regarded for retransmission only small chunks of the missing segments can be retransmitted (the black boxes in Figure 7 only appear as boxes due to the low resolution of the plot). The problem of the restricted algorithm is shown in more detail in the zoomed out part of Figure 7 that represents an enlarged part of the out-of-band FSC simulation for this algorithm. The high frequency of layer changes will be very annoying for the client who is currently watching the video. In contrast to the restricted algorithm the result of the unrestricted algorithm is that the quality of the video is enhanced by one layer in one contiguous segment which in this specific case does not lead to additional layer changes compared to the initially cached video. The difference in the amount of retransmitted segments in comparison to the in-band simulation (see Figure 6) is caused by the fact that the amount of spare bandwidth that is available for retransmission is higher in the in-band case.

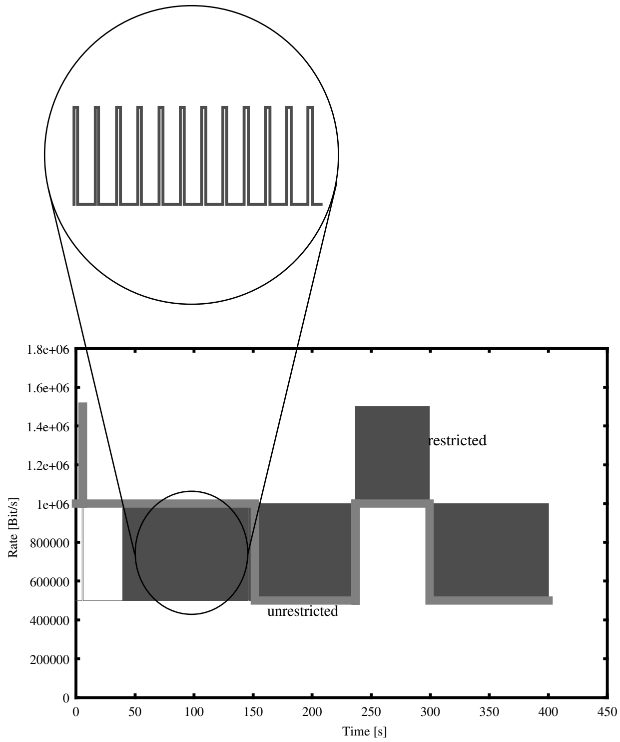

Figure 7: Out-of-band FSC simulation

3.3.3 Influence of number of layers

It is obvious that the amount of bandwidth that is available for retransmission will decrease with an increasing number of layers caused by a higher adaptiveness to the bandwidth determined by TFRC. With this simulation we wanted to investigate the dependency between the number of layers of a layer encoded video and the resulting amount of bandwidth for retransmissions. In the simulation we always used the same TFRC trace and varied the number of layers between 2 and 20. Increasing the number of layers does not increase the maximum bandwidth of the layer encoded video, rather the bandwidth of each single layer is decreased. This had to be done to be able to compare the results of each single simulation. Figure 8 presents the result of the simulations which shows the percentage of the overall capacity of the TFRC trace that can be used for retransmissions. The result states our assumption that the additional capacity will decrease with an increasing layer granularity. But even with a high number of layers there is still capacity for retransmissions available.

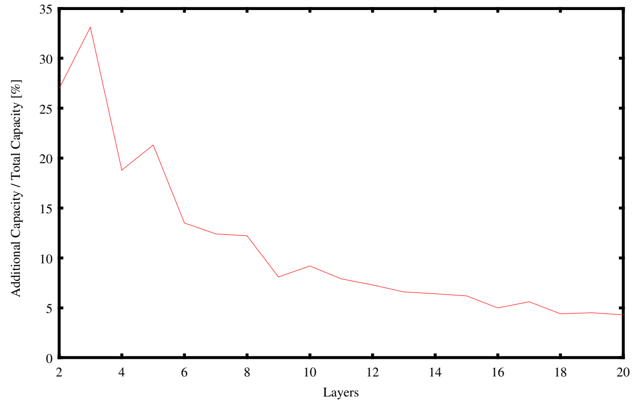

Figure 8: Relative additional capacity for retransmissions

4 IMPLEMENTATION DESIGN FOR FSC

In this section, we present the design for an implementation of the FSC technique which should be based on already existing, standardized and if possible well established protocols and techniques. However, that has not always been possible which is mainly due to the fact that the proposed TCP-friendly mechanisms requires some significant changes of the protocol behavior. Nevertheless, the design should only require modifications at origin server and proxy cache to allow standard clients in this architecture. In general, we distinguish between the transmission of time critical data (the actual stream that is transmitted) and time uncritical data (segments for retransmission). It is the overall goal of this section to show that FSC can be reasonably integrated in streaming applications.

4.1 Protocol Suite

The most common approach for audio and video streaming is the usage of RTP[‡] over UDP as transport protocols. It is well known that this approach lacks an appropriate congestion control mechanism and might cause problems like congestion collapse if the amount of audio and video streams further increases. That is exactly why different variations of TCP-friendly protocols have been developed that should avoid the occurrence of such problems in the best-effort Internet. As mentioned before TFRC is one of these protocols and in Section 3.1 we already stated why we favor it as a protocol for streaming environments. Another advantage is that the TFRC mechanisms can be integrated into the RTP protocol and thus the introduction of a completely new protocol in the streaming protocol suite is not necessary. This integration has the additional benefit that no modifications to UDP must be made and therefore possible kernel modifications can be avoided. To enable TFRC functionality in RTP some new header informations are needed (see Figure 9) and part of the overall protocol behavior must be changed. Fortunately two of the additionally needed header fields are already contained in the RTP header, *sequence number* and *time stamp,* respectively. The additional fields shown in Figure 9 must be put in the RTP extension header. The receiver reports needed by TFRC can be transported by the *application specific information* in the RTCP receiver reports. The frequency of RTCP receiver reports must be highly increased since TFRC requires to send these reports every RTT. Since we envision only unicast transmission so far in our architecture the higher amount of reports should neither restrict the raw data transmission nor cause an ACK implosion.

Rather complicated is the identification of missing segments of a layer encoded video that should be retransmitted. One could imagine that missing segments could be easily identified by the RTP sequence number but this is not true in every case. The sequence number of an RTP packet would only be helpful if the data would be stored as RTP packets on the origin server's disk, because the simple information of a sequence number would not be sufficient to identify the related part of, e.g., a file that contains an MPEG-1 video where the packet length can vary (wire format and storage format must not necessarily be identical). In the case of LC-RTP the loss recognition is realized by a *byte count* which is included in each RTP header. The byte count represents the actual byte position of the data that is included in the RTP packet. Each 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 server implementation stores the file as raw data and another stores some header information with it. 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. With the aid of the byte count losses can be exactly identified, the

‡. For reasons of simplicity we only mention RTP but always mean RTP and RTCP.

Sender
sequence number*
time stamp*
round trip time
bitrate
round
mode

*already included in the RTP header

Receiver
sequence number
time stamp
b_rep
b_exp

Figure 9: Additional TFRC header fields

receiver can maintain a list of losses and the lost segments can be requested from the sender at another point in time. If lost segments should be retransmitted during the streaming session the RTCP application specific header can be used to send the loss lists from the receiver to the sender. Should the retransmission be performed out of band, a TCP connection would be sufficient to transmit the loss lists to the sender. Since there exist now two cases that require an RTP extension header we propose that in the case of FSC the RTP protocol should be used with an modified extension header as shown in Figure 10.

```
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              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        rount trip time                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            bitrate                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                      loss measurement round                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             mode                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          byte count                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          byte count                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 10: RTP header extension

An additional issue is the multiplexing of the video stream and retransmitted segments over one RTP session. In this case we can make use of RTP's mixing functionality. Originally this functionality was thought to combine RTP streams from different senders at a router into one RTP stream. We make use of this functionality in a slightly different way: In our scenario no physically separated senders exist but the layer encoded video and the packets that should be retransmitted can be regarded as two logical sources. Thus both streams[**] can be transmitted via one RTP stream whereas each stream is assigned a different *synchronization source identifier* (SSRC). This technique allows the RTP receiver to correctly identify each of the two streams and forward the packets to their correct destination. It might also be possible to mix more than two streams with this mechanism but this is out of the scope for this work. To identify each SSRC correctly the receiver needs additional information about the mapping between streams and SSRCs. The mapping information can be signaled to the receiver with the aid of the *private extension source description* (SDES) item of an RTCP source description packet. This type of RTCP packet contains a list of SSRCs and according SDES items. The *private extension* item is meant for experimental or application specific use. The SDES *private extension* consists of an *item identifier*, *length information*, *prefix length*, *prefix* and *value string* (see Figure 11).

**. To simplify description the retransmission of segments is also described as a stream, although this is not technically correct.

```
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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     PRIV=8    |    length     | prefix length | prefix string ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
...             |              value string                    ...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Figure 11: Private extension SDES item

The *prefix string* for this specific SDES item will be set to **FSC** to indicate that this information is related to the fair share claiming technique. For the *value string* three different strings are defined so far:

Value string	Description
STREAM	SSRC represents layer encoded video stream
INBAND	SSRC represents a stream for retransmitted segments that belong to the in parallel streamed layer encoded video
OUTBAND	SSRC represents a stream for retransmitted segments that belong to an already cached video

Table 1: Value string parameters

This additional information allows the demultiplexing of the single sessions of an RTP stream and their correct assignments to instances for further data processing. In Section 4.3 we will present how a correct data path could be established with the aid of our *Stream Handler* (SH) [24] architecture.

4.2 Retransmission Signaling

As mentioned above it might be possible to perform in-band and out-of-band retransmission with FSC. With out-of-band retransmission the respective video is already stored on the cache and one run of the retransmission scheduling algorithm should be sufficient to generate a retransmission list. A simple TCP transmission from the receiver to the sender to send the list of ordered missing segments should be sufficient. The sender stores this list and in the case of a retransmission request uses this list to obtain information which segments of the original video should be retransmitted.

In the case of in-band retransmission the retransmission signaling must be handled in a different way. First of all the video is not entirely transmitted to the cache. The retransmission scheduling algorithm can only make decisions based on the already received part of the video. Thus the generated list of segments that should be retransmitted might change over time and updates of the list that exists at the sender must be performed. To be able to perform this update we propose that the initial list that is created by the retransmission scheduling algorithm is also stored on the receiver. Each time the algorithm is performed again, the newly generated list and the stored list should be compared. If the differences reach a certain threshold value (for a suitable metric that measures similarity between loss lists) a new list must be transmitted to the sender. The determination of the threshold value for retransmission list updates goes beyond the scope of this paper and is an issue for further research.

4.3 Stream Handler Extension

Our experience with the implementation of streaming applications showed us the need for a generic architecture to handle continuous media streams. This became specifically clear during the development of our experimental KOM-Player platform [24]. The platform is used for investigations on AV distribution systems and therefore has to offer support for different encoding formats, transport protocols, but also distribution mechanisms under investigation. Such distribution mechanisms may combine unicast and multicast distribution or may apply segmentation and reordering for efficient delivery. During the initial implementation phase we quickly realized that a monolithic approach would not allow a simple integration of these new distribution mechanisms. This led to our decision to build an environment that is based on a stream handler (SH) architecture.

In this section we will define the extensions that must be made in the stream handler architecture to support the FSC technique. In our experimental VoD platform we developed server, proxy cache and a client all of which make use of the stream handler architecture. Therefore, a great deal of stream handler modules have already been designed, implemented and tested. To support FSC in our platform we try to reuse these elements and if necessary extend them or create new stream handlers. First of all we show the modifications that have to be made to the already existing SHs and then we will present the design of the new SHs.

4.3.1 RTP/RTCP and LC-RTP/LC-RTCP:

The RTP and RTCP functionality is combined in the RTP SH. The FSC technique requires extensions and modifications to both RTP and RTCP as described in *Section 4.1*. For the multiplexing and demultiplexing functionality the RTP SH must be able to receive data from more than one upstream SH (sender) and forward it to more than one down stream SH (sender) in the case of out-of-band retransmission. This is not the case for in-band retransmission, since the byte count information of LC-RTP defines the position of the retransmitted segment explicitly. Although only LC-RTP is used in the FSC case, the extension should be made for both protocols since a) RTP is a basis for LC-RTP and b) the new functionality might also be needed by RTP only. This will allow a separated development of RTP with TFRC mechanism and LC-RTP.

On both the sending and receiving parts, extension to provide TFRC functionality must be made, too. New fields must be added to the extension header as depicted in Figure 10. The changes for RTCP are extensive since on one hand the format of the RTCP receiver reports and on the other hand the timing for the transmission of these must be changed. The TFRC specific information should be transported in application-specific information of an RTSP receiver report and should contain the four fields shown in Figure 9. The timing for the receiver reports must be changed in a way that it is based on the RTT information instead of the algorithm proposed in [20].

4.3.2 Packetizer and depacketizer

Several profiles for the transport of standardized audio and video formats in RTP exist [19]. So far no profile for the transport of layer encoded video is defined, caused by the lack of a standard for this technique. Our experience with the development of (de-)packetizers for several audio and video formats [24], have shown that building new SH for this purpose is a rather simple task. Depending on what layer encoded video techniques should be supported it might be necessary to build more than one SH. One of our future tasks is to investigate layer encoded video proposals with respect to the needs of our system. If this work is done we will be able to decide which of the proposals should be supported and develop the appropriate packetizer and depacketizer SHs. To describe the interaction between the SHs in more detail an example scenario for in-band retransmission is shown in Figure 12. Whenever a RTP packet should be sent out on the network the payload will be pulled from the Packetizer SH. With the report functionality of the SH architecture the RTPSink SH can inform the Packetizer SH about the actual transmission rate. With the aid of this information the packetizer can build the two different types of payloads. For the layer encoded video the rate information will also determine the number of layers that should be transmitted. If the number of layers is known the packetizer can determine the resulting capacity that is available for the retransmission of segments and determine which type of packet (layer encoded video or retransmitted segment) will be handed to the RTPSink SH in the case of a pull request from the latter. With each payload the packetizer must also provide the appropriate SSRC information which will allow the correct demultiplexing in the RTPSource SH at the receiver.

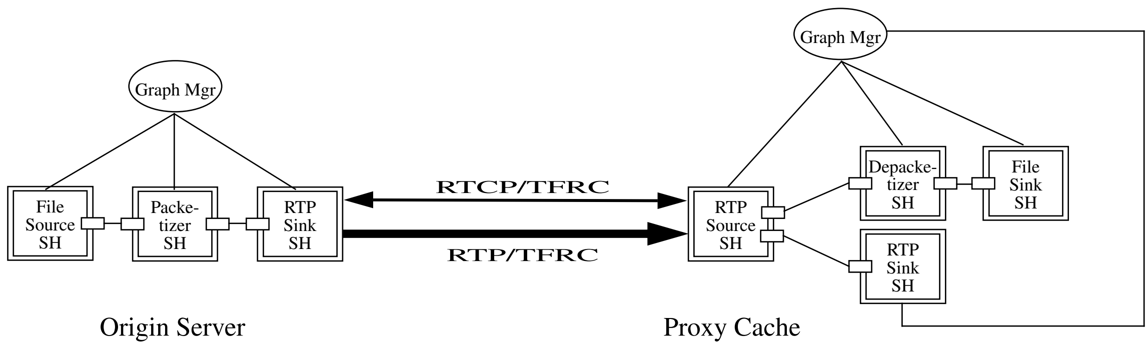

Figure 12: Stream handlers scenario for in-band FSC

5 RELATED WORK

Rejaie et al [15] and Saprilla et al [18] also present mechanisms that claim their fair share and support the transport of layer encoded video. Both assume that the client has sufficient buffer to allow a transmission rate higher than the receivers consumption rate. While the first approach is limited to CBR encoding the second also supports VBR layer encoding. In contrast to the FSC mechanism we presented in this paper video transmission into proxy caches is not considered. Both mechanism do not support the transmission of data that has already missed its deadline for the timely consumption at the client and therefore do not offer any functionality to improve the quality of a video that is being cached or already stored on a cache.

Another approach that supports scalable streams is presented by Law et al [8]. In this work the focus is mainly on server efficiency and scalability. In comparison to our approach the quality is adapted due to the capabilities of the receiving client rather than to network conditions. Their architecture does neither envision caches nor incorporate TCP-friendly streaming.

[7] considers the combination of caching and layered video, yet, the latter only for the support of heterogeneous clients but not for congestion control purposes. Furthermore, the emphasis of their work is on optimal cache replacement decisions viewed over *all* videos stored in a cache. We, however, assume a two-stage decision process where in the first stage a video is selected for storage in a cache and then the retransmissions of missing segments are scheduled independent from the cache status of other videos. While this represents a restricted problem it ensures that the overall problem still remains manageable. Another difference in their work is the fact that missing segments of a certain layer a only streamed directly to the client in contrast to our approach where the segments are transmitted to the proxy cache to achieve a quality improvement for more than one client.

The work of Nelakuditi et al [12] and Rejaie et [17] is mainly concerned with quality improvement of layer encoded videos and does not consider a fair share claiming technique. [12] present an approach where only server and clients are involved and therefore the client requires sufficient buffer space to allow quality improvement of layer encoded video. Closer to our work is the approach presented in [17] where also missing segments of a layer encoded video are retransmitted to proxy caches in order to improve the quality of the cached videos. However, we extend their work by focussing on the development and comparison of different retransmission scheduling algorithms which are more flexible and performing better than the one presented in [17] as shown in Section 3.3.2. In contrast to our FSC algorithm an additional link for retransmission is provided between the origin server and the proxy cache.

An architecture of video servers, caches and clients for layer encoded video is proposed in the work of Paknikar et al [13]. In contrast to our SAS proposal a single broker exists that handles all client requests and redirects them to the corresponding cache. Even though the usage of a broker allows to reduce the complexity of the caches it has the disadvantage that in the case of a broker failure the clients will not be able to request content.

Fine-granular-scalable video [14] that consists of a base layer and one enhancement layer which is coded in a way to enable the transmission of video at any desired bitrate is very well suited for FSC. However it has not been shown (to our best knowledge) that this proposed technique is usable in streaming applications since some of the encoding has to be performed in real-time.

6 CONCLUSION AND FUTURE WORK

In this paper, we presented a new technique called FSC that allows to claim the fair share for TCP-friendly sessions for the transmission of layer encoded videos in the case that proxy caches are involved. This technique bears the advantage that on one hand these sessions actually will get their fair share of the link and on the other hand the quality of already cached videos can be improved. We presented our overall scenario of scalable adaptive streaming and showed how FSC fits into it. In order to prove the applicability of FSC we created a simulation environment that consists of three single simulation steps: Creation of TFRC traces, layer encoded video transmission and retransmission scheduling. A series of simulations based on this simulation environment were performed. The results of the simulations stated the applicability of FSC, especially, that in combination with the retransmission scheduling algorithms we developed earlier a reasonable quality improvement for already cached videos can be achieved. We have also shown that an other already existing retransmission scheduling algorithm is not well suited for the proposed FSC technique. Additionally, we have shown how the proposed FSC technique can be integrated in streaming applications by the extension of already existing protocols.

The approach is so far limited to unicast transmissions. In future work we will investigate how also multicast transmission can be supported with the FSC approach. In the next step we will implement the presented design in our streaming platform to be able to gain some information about how FSC performs in a “real world” scenario. An other interesting approach that we would like to compare with FSC an investigate in more details is the one presented in [12] with the exception that in our case proxy caches will be involved and the reordering an buffering will be performed at the proxy cache to keep the clients as simple as possible.

7 REFERENCES

[1] Bob Braden, David Clark, and Scott Shenker. Integrated Services in the Internet Architecture. Internet RFC 1633, June 1994.

[2] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings of the ACM SIGCOMM '00 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication 2000, Stockholm, Sweden*, pages 43–56, August 2000.

[3] Ron Frederick, Jay Geagan, Mike Kellner, and Alagu Periyannan. Caching Support in RTSP/RTP Servers. Internet Draft, March 2000. Work in Progress.

[4] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, C.J. Sreenan, and S. Wen. A Simple Loss Differentiation Approach to Layered Multicast. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies 2000, Tel-Aviv, Israel*, pages 461–469, March 2000.

[5] Carsten Griwodz. *Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure*. PhD thesis, Darmstadt University of Technology, Darmstadt, Germany, April 2000.

[6] Carsten Griwodz, Michael Bär, and Lars C. Wolf. Long-term Movie Popularity in Video-on-Demand Systems. In *Proceedings of ACM Multimedia'97*, pages 340–357, November 1997.

[7] Jussi Kangasharju, Felix Hartanto, Martin Reisslein, and Keith W. Ross. Distributing Layered Encoded Video through Caches. In *Proceedings of the Tenth Annual Joint Conference of the IEEE Computer and Communications Societies 2001, Anchorage, NY, USA*, April 2001.

[8] Kelvin K. W. Law, John C. S. Lui, and Leana Golubchik. Efficient support for interactive service in multi-resolution vod systems. *VLDB Journal*, 8(2):133–153, 1999.

[9] Jae-Yong Lee, Tae-Hyun Kim, and Sung-Jea Ko. Motion Prediction Based on Temporal Layering for Layered Video Coding. In *Proceedings ITC-CSCC'98*, pages 245–248, July 1998.

[10] Thomas D.C. Little and Dinesh Venkatesh. Prospects for Interactive Video-on-Demand. *IEEE Multimedia*, 1(3):14–25, May 1994.

[11] Sean McCreary and kc claffy. Trends in Wide Area IP Traffic Patterns, May 2000. http://www.caida.org/outreach/papers/AIX0005/.

[12] Srihari Nelakuditi, Raja R Harinath, Ewa Kusmierek, and Zhi-Li Zhang. Providing Smoother Quality Layered Video Stream. In *Proceedings of the 10th International Workshop on Network and Operating System Support for Digital Audio and Video, Raleigh, NC, USA*, June 2000.

[13] Shantanu Paknikar, Mohan Kankanhalli, K.R. Ramakrishnan, S.H. Srinivasan, and Lek Heng Ngoh. A Caching and Streaming Framework for Multimedia. In *Proceedings of the ACM Multimedia Conference 2000, Los Angeles, CA, USA*, October 2000.

[14] Hayder Radha and Yingwei Chen. Fine-Granular-Scalable Video for Packet Networks. In *Proceedings of the Packet Video'99 Conference , New York, NY, USA*, April 1999.

[15] Reza Rejaie, Mark Handley, and Debora Estrin. Quality Adaptation for Congestion Controlled Video Playback over the Internet. In *Proceedings of the ACM SIGCOMM '99 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication 1999, New York, NY, USA*, pages 189–200, August 1999.

[16] Reza Rejaie, Mark Handley, and Deborah Estrin. RAP: An End-to-End Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies 1999, New York, NY, USA*, pages 395–399, March 1999.

[17] Reza Rejaie, Haobo Yu, Mark Handley, and Debora Estrin. Multimedia Proxy Caching for Quality Adaptive Streaming Applications in the Internet. In *Proceedings of the Nineteenth Annual Joint*

Conference of the IEEE Computer and Communications Societies 2000, Tel-Aviv, Israel, pages 980–989, March 2000.
[18] Despina Saparilla and Keith W. Ross. Optimal Streaming of Layered Video. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies 2000, Tel-Aviv, Israel*, pages 737–746, March 2000.
[19] Henning Schulzrinne. RFC 1890 - RTP Profile for Audio and Video Conferences with Minimal Control. Standards Track RFC, January 1996.
[20] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RFC 1889 - RTP: A Transport Protocol for Real-Time Applications. Standards Track RFC, January 1996.
[21] Ke Shen and Edward J. Delp. Wavelet Based Rate Scalable Video Compresssion. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):109–122, February 1999.
[22] Joerg Widmer, Robert Denda, and Martin Mauve. A Survey on TCP-Friendly Congestion Control. Technical Report TR-2001-002, Universitaet Mannheim, February 2001.
[23] Michael Zink, Carsten Griwodz, Alex Jonas, and Ralf Steinmetz. LC-RTP (Loss Collection RTP): Reliability for Video Caching in the Internet. In *Proceedings of the Seventh International Conference on Parallel and Distributed Systems: Workshops*, pages 281–286, July 2000.
[24] Michael Zink, Carsten Griwodz, and Ralf Steinmetz. KOM Player - A Platform for Experimental VoD Research. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*. IEEE Computer Society Press, July 2001.
[25] Michael Zink, Jens Schmitt, and Ralf Steinmetz. Retransmission Scheduling in Layered Video Caches. Technical Report TR-KOM-2001-07, Darmstadt University of Technology, February 2001.