An Adaptive MPEG-4 Proxy Cache

Peter Schojer
Laszlo Böszörményi
Hermann Hellwagner

UNIVERSITÄT KLAGENFURT

Institute of Information Technology
University Klagenfurt
Technical Reports
No TR/ITEC/02/2.07, June 2002

An Adaptive MPEG-4 Proxy Cache

Peter Schojer
Laszlo Böszörményi
Hermann Hellwagner

UNIVERSITÄT KLAGENFURT

Institute of Information Technology
University Klagenfurt
Technical Report No TR/ITEC/02/2.07
June 2002

Abstract

Multimedia is gaining ever more importance on the Internet. This increases the need for intelligent and efficient video caches. Typical Web proxies were not designed to efficiently support the caching of videos. A promising approach to improve caching efficiency is to adapt videos. With the availability of MPEG-4 it is possible to develop a standard compliant proxy that allows fast and efficient adaptation.

We propose a modular design for an adaptive MPEG-4 video proxy that supports efficient full and partial video caching in combination with filtering options that are driven by the terminal capabilities of the client. We use the native scalability operations provided by MPEG-4 and use the emerging MPEG-7 standard to describe the scalability options for a video. The proxy parses the MPEG-7 description and decides, based on this description and the terminal capabilities of the client, which adaptation step to choose. Simple MPEG-4 audio-visual streams are supported by filter operations in the compressed domain that realize several temporal scaling algorithms and color reduction. In this paper, we will restrict ourselves to full video caching.

The combination of adaptation with MPEG-4, MPEG-7 and client terminal capabilities is to the best of our knowledge unique and will increase the quality of service for end users.

Keywords: adaptation, MPEG-4, MPEG-7, adaptive proxy, caching

Contents

1 Motivation

The importance of multimedia on the Internet is steadily increasing. A new dimension of complexity is added to Web proxies by the heterogeneity found in end terminals. Adaptation can be used to improve caching efficiency and to support new end terminals. On the one hand, the proxy can be used as a media gateway to *adapt* videos to fit to the needs of a client. On the other hand, similary to usual Web proxy caching, storing popular videos close to the clients will reduce network/server load and start-up latency.

In this paper we show the design of an adaptive Web proxy using RTSP and the multimedia standards MPEG-4 and MPEG-7.

Section 2 gives a short overview of related work. Section 3 explains the idea of adaptation and also the concepts behind MPEG-4 and MPEG-7. Section 4 shows how these concepts are used to realize an adaptive MPEG-4 proxy. Section 5 explains in detail a possible client-proxy-server interaction scenario. In Section 6 we give a short conclusion and an outline of future work.

2 Related Work

There has been few practical work carried out in the area of adaptation of cached, layered encoded videos.

In [5] an adaptive proxy is introduced. The corresponding implementation is based on a proprietary AT&T client-server architecture, where little public information is available.

[7] proposes a proxy that filters GOPs (Group of Pictures) based on the available bandwidth; only simulation results are presented. [8] proposes a way to deal with retransmission of missing segments of layered encoded videos; again simulations are used to derive results. [4] performs an exhaustive comparison of replacement strategies suitable for layered encoded videos, again based on simulation results. [6] proposes a pre-fetching mechanism for optimizing the quality hit rate in a proxy (a metric considering the quality of an adapted video compared to the quality of the original video).

While there is nothing wrong with simulation when evaluating new video cache replacement strategies, simulation reaches its limits, if one wants to analyze how a CPU intensive task like an adaptation step effects the overall performance/scalability of the proxy.

To the best of our knowledge, we are actually the first who present a design and an implementation (still in progress) for an adaptive MPEG-4 proxy that considers terminal capabilities and MPEG-7 meta-information.

3 Background

3.1 Adaptation

Media-adaptation is the ability of a system to change the quality of a multimedia stream according to available resources [9]. In this work, we restrict adaptation to non-linear quality reduction. Non-linearity in this context means that we gain a large size reduction for a relatively moderate quality loss.

In [4] several cache replacement strategies for layered videos have been evaluated, clearly showing that using adaptation in proxies not only results in a higher object hit rate but also in a higher quality-weighted hit rate (a metric combining object hit rate and quality hit rate).

A special variant of adaptive proxies are *media gateways* that allow low-end devices to view a down-scaled version of the video. Media gateways additionally support transcoding of videos, a CPU intensive task.

The integration of terminal capabilities and meta-information enables our proxy to do both adaptation for the sake of a higher hit rate and to act as a simple media gateway, doing efficient adaptation in the compressed domain.

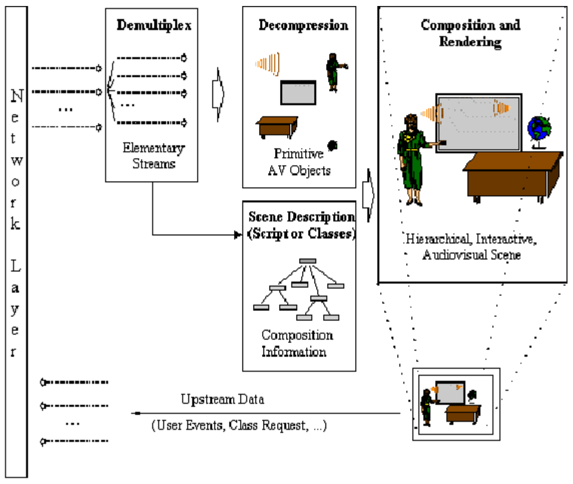
As already indicated previously, adaptation steps depend on the type of videos used, e.g. resizing MPEG-1 videos requires a CPU intensive recompression of the video. Fortunately, with MPEG-4 a standard is available that provides support for adaptation.

3.2 MPEG-4

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group), which became an International Standard in the first months of 1999. Compared to older MPEG standards it is the first that offers the possibility to structure audio-visual streams. MPEG-4 defines the notion of a scene (see Figure 1) relying on similar notions of virtual reality standards. A scene consists of a set of media objects, each having a position, a size and a shape [1]. Each media object consists of at least one *Elementary Stream* (ES). If more than one ES is present for an visual object, the object offers native adaptation support. The additional ESs are used to add spatial, temporal, or SNR scalability to an object. Fast adaptation happens by simply dropping an ES, which is equivalent to deleting a file in the proxy.

An ES itself consists of a sequence of *Access Units* (AU), where usually one AU encapsulates one frame. To transfer the AU over the network, the AU is packetized to *SyncLayer packets* and then passed to the delivery layer, where it is packetized to an RTP packet. Depending on the size of the AU, one or more SL packets are needed [1].

An MP4 movie (with video & audio) consists of at least five ESs. One is the *Initial Object Descriptor* (IOD) stream that keeps references to the objects used in the video. The objects are described in the *Object Descriptor* (OD) stream. The third mandatory stream



Network Layer
Demultiplex
Elementary Streams
Decompression
Primitive AV Objects
Composition and Rendering
Hierarchical, Interactive, Audiovisual Scene
Scene Description (Script or Classes)
Composition Information
Upstream Data
(User Events, Class Request, ...)

Figure 1: Example for an MPEG-4 Scene [1]

is the BIFS stream (Binary Information For Scenes), that stores the scene description, actually telling the decoder which object is visible at which point in time and where exactly it should be rendered. Optionally, one can have several ESs for the video and one for the audio [1].

While MPEG-4 offers support for adaptation, the question remains how the proxy determines which adaptation step will give the most benefit in a certain situation. This can only be solved by adding meta-information to a video, as defined by MPEG-7.

3.3 MPEG-7

MPEG-7 is another ISO/IEC standard and aims at describing multimedia data. With MPEG-7 one can add semantic information to a video. For the proxy, meta-information regarding adaptation is especially important. We restrict ourselves to the content adaptation part of MPEG-7, for further details see [2].

MPEG-7 features a *Description Definition Language* (DDL), that allows one to generate a *Description Schema* (DS), which in turn is used to code *Descriptors*.

The VariationRelationship descriptor of the Variation DS is used to describe a possible adaptation step. There are several descriptors defined, like *Summarization, Extraction, ColorReduction, SpatialReduction*, or *QualityReduction*, to name just a few.

Figure 2 shows an example for a *colorReduction* variation, that - if applied to the 30 MB video *test.mpg* - would create a new video *out.mpg* with a fidelity of 0.8 (20% quality loss) and a resulting size of 15 MB. The priority field is used by the proxy to choose a variation from a VariationSet. Unfortunately, the current version of MPEG-7 doesn't support the specification of resource consumptions [3].

```
<Mpeg7>
  <Description xsi:type="VariationDescriptionType">
    <VariationSet>
      <Source xsi:type="VideoType">
        <Video>
          <MediaLocator>  <MediaUri>file://test.mpg</MediaUri>  </MediaLocator>
          <FileSize>30000000</FileSize>  <BitRate> 512000</BitRate>
        </Video>
      </Source>
      <Variation fidelity="0.8" priority=1">
        <Content xsi:type="VideoType">
          <Video>
            <MediaLocator>  <MediaUri>file://out.mpg</MediaUri>  </MediaLocator>
          <FileSize>15000000</FileSize>  <BitRate> 256000</BitRate>
          </Video>
        </Content>
        <VariationRelationship>colorReduction</VariationRelationship>
      </Variation>
    </VariationSet>
  </Description>
</Mpeg7>
```

Figure 2: Example of an MPEG-7 variation descriptor

4 Adaptive Proxy

4.1 Design

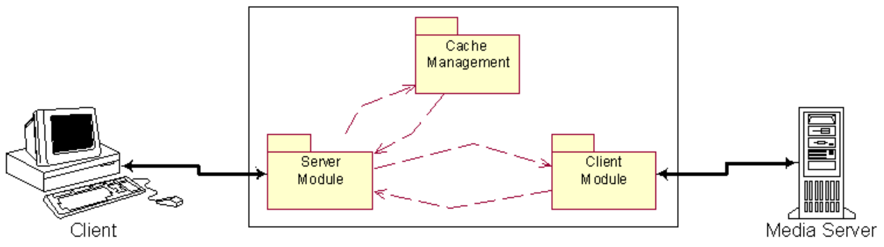


Figure 3: Proxy Architecture

A proxy is inserted into the path between client and server. It has to "imitate" a server for the client and a client for the server. Thus, the design of the proxy consists of three main parts: a server module, that implements server functionality dealing with client requests; a client module that is responsible for establishing a connection with the media server; and a cache manager that manages cached objects and realizes cache replacement strategies. Figure 3 shows how these parts communicate with each other. The server part uses the cache manager to lookup a video and to insert videos into the cache. The client part is used to retrieve missing videos/ESs from the server.

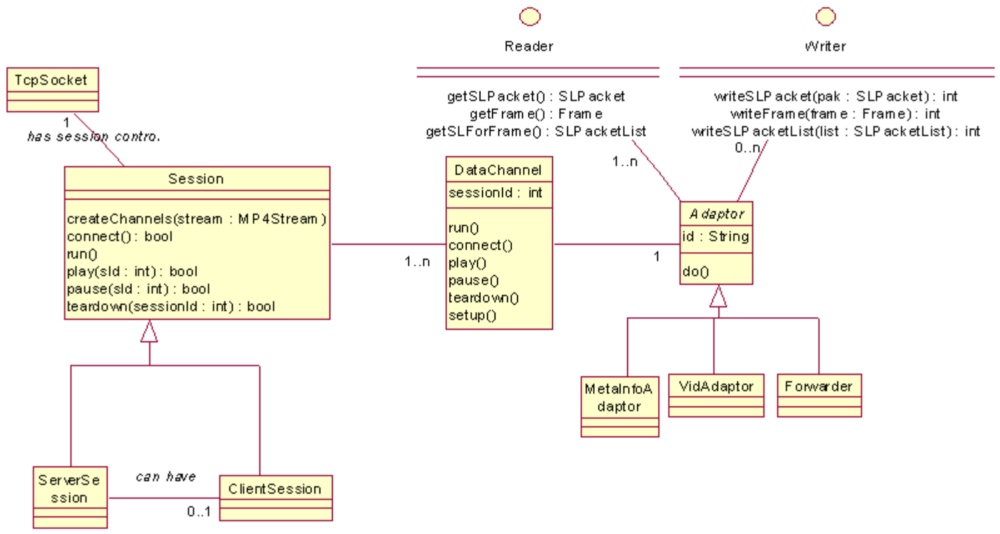


Figure 4: Server Module

4.2 Server Module

The server module listens for RTSP requests and manages existing sessions. For a new request a *ServerSession* object is created (see Figure 4). If the proxy encounters a partial hit, i.e. it has to fetch missing ESs from the media server, a *ClientSession* object is needed too. Each session object listens at a TCP socket for RTSP commands and forwards them to their corresponding *DataChannels*. A DataChannel is the link between the media data and the client and encapsulates an *Adaptor*. An Adaptor reads data from a *Reader*, which could be a network or a file reader, adapts internally the data - for example a TemporalAdaptor could drop all packets transporting B-frames -, and then outputs the result to one or more *Writer* objects. In the case of a cache miss for an ES, the Client Session object will create an adaptor with a network reader and two Writer objects: a file writer for caching and a network writer for forwarding. A simplified data flow is shown in Figure 5. To keep the example concise, the data flows writing the data to the disk are omitted.

4.3 Adaptation in the Proxy

Adaptation is part of the Server module. It supports the following:

- ES Dropping

  This decision happens during the RTSP DESCRIBE phase, where an SDP (Session Description Protocol) description is generated, which contains the IOD stream and a simple description of all ESs of a video (like average bit rate of a stream, its stream-id, and whether a stream is audio or video or a generic stream like BIFS). Removing

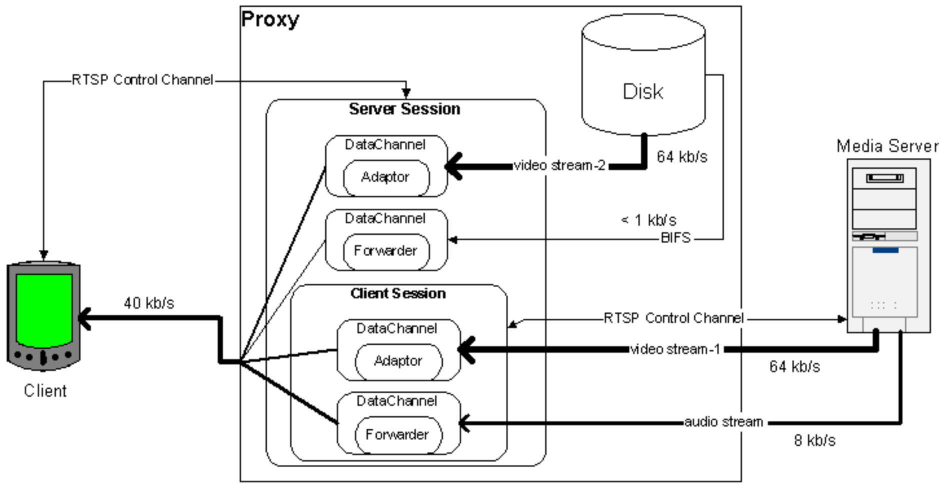


Figure 5: Data flow with adaptation

ESs from the SDP description realizes - if the media supports it - *temporal*, *spatial* and *SNR scaling*.

- ES Adaptation

  Whether to adapt an ES, is decided during session creation where the Adaptors are set at the DataChannels. If no adaptation is required, a *Forwarder* object is created that simply forwards data from the Reader to the Writers. This decision depends on the terminal capabilities and the MPEG-7 meta-information associated with the MPEG-4 stream. Adaptors work on a single ES. Currently, the following adaptation classes are supported:

  – temporal scaling (B-frame dropping, GOP dropping, extraction of key-frames) and

  – color reduction (grey-scaling)

  All of these operations work in the compressed domain. Operations that require CPU intensive decompression are currently not supported.

4.3.1 Meta-Information

To adapt a video efficiently, in-depth information about the video is needed, such as which ES is enhancement layer (and thus can be removed) and which is base layer.

This information can be extracted from the OD stream and is needed rather early during the RTSP DESCRIBE phase. In the case of a complete cache miss, one has to download the (very small) OD stream completely and parse this stream to detect these dependencies. The SDP description received from the server can be used to determine the bit rate of an individual stream and hide ESs from the client accordingly, simply by removing them from the SDP description. This is only a rough and defensive estimation based on average bit rate values. To determine the total bandwidth requirement the BIFS stream must be parsed - but this intelligence is better integrated at the client.

To determine if an adaptation step is allowed, one has to parse the MPEG-7 stream for VariationRelationship descriptors and pick the one that best fulfills the requirements of the client. This stream is also needed in advance.

Additionally, the meta-information has to be updated to reflect the changes (lower bit rate, lower frame rate, update/delete variation descriptors), to avoid that another proxy applies the same adaptation. More information on the use of meta-data can be found in [9].

4.4 Cache Management

The main functionality of the cache management module is to store the received data and to realize cache replacement algorithms. Within the proxy, an ES is modelled as a

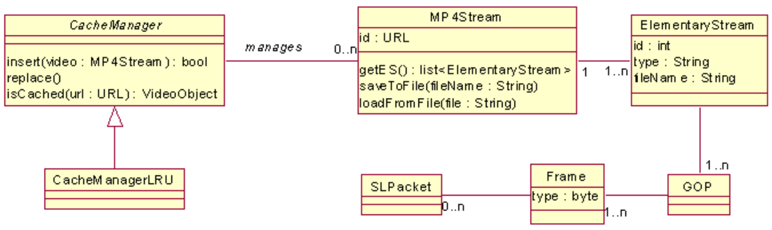


Figure 6: Cache Management

sequence of GOPs, which consists of frames, which are stored as a sequence of SyncLayer [1] packets (Figure 6). The *CacheManager* class itself is defined as an abstract class, allowing several cache replacement strategies to be realized.

5 Example

Figure 7 shows three possible scenarios that can occur in an adaptive proxy: a complete cache miss, a partial cache hit and a complete cache hit. All three scenarios are simplified; a video consists only of two visual streams. *SDP*– or *data*– means that an adapted version is sent to the client. We will explain a partial cache hit in detail. Suppose that video-1 consists of 8 ESs (1 IOD, 1 OD stream, 1 BIFS, 1 audio stream, 2 video objects, each consisting of 2 ES). The proxy has already seen video-1 and has reduced each video object to its base layer.

- The client starts by sending its terminal capabilities to the proxy.
- The client then sends an RTSP DESCRIBE request.

```
DESCRIBE rtsp://127.0.0.1/videoandaudio.mp4 RTSP/1.0
CSeq: 1
```

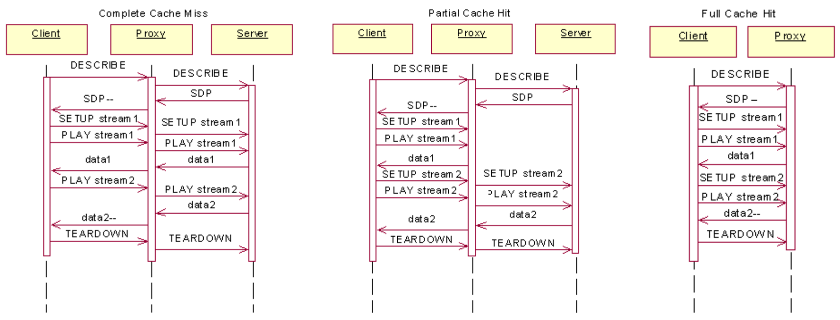


Figure 7: Caching Scenarios

- At the proxy, DESCRIBE triggers the creation of a currently empty *ServerSession* object associated with the RTSP port to the client. The *CacheManager* module is asked, if for this URL any ESs are cached. Based on this information and the terminal capabilities of the client, an SDP description is created and sent to the client. Additionally, the ServerSession object starts to create *DataChannels* (DC); for each cached ES one *DC* reading from a local file, is created. If one ES is missing, the ServerSession object has to create a *ClientSession* object, that has an RTSP connection to the media server and adds *DCs*, which read from the network, to the ClientSession object. Note that the ClientSession object immediately forwards the DESCRIBE command to the server.

  If the terminal capabilities file indicates that the end-device can not handle all ESs, some ESs are simply omitted from the SDP description. In the example, this is the enhancement ES from video object 2.

  If the terminal capabilities file and the pre-cached MPEG-7 meta-data indicates that the client could handle an adapted version of the video, a Filter/Adaptor is set at the affected DCs.

- The client can now parse the SDP description and extract the number of ESs. An example for such a reply including IOD and one video-ES description is

```
RTSP/1.0 200 OK
CSeq: 1
Content-Type: application/sdp
Content-Length: 1294
v=0
o=- 1018430638 194857 IN IP4 127.0.0.1
s= RTSP Session
a=mpeg4-iod "data:application/mpeg4-iod;base64,AoCAgHAADwEBAgEBA4CAgDU [...] "
m=video 0 RTP/AVP 90
a=control: rtsp://127.0.0.1/videoandaudio.mp4/streamid=2115
a=mpeg4-esid 2115
a=fmtp:90 StreamType=4; SizeLength=16; DTSDeltaLength=8;
config=0480808021C211034BC00000000000000000000058080800F6F62736F6C65746520737472696E67
a=rtpmap:90 mpeg4-generic
```

- For each ES, an RTSP SETUP request is generated:

  SETUP rtsp://127.0.0.1/videoandaudio.mp4/streamid=2115 RTSP/1.0

  CSeq: 6

  Transport: RTP/AVP/UDP;unicast;client_port=50002-50003

- The proxy sees 6 SETUP requests, which are forwarded to their corresponding DCs. The OD, BIFS and audio streams are cached, thus their DCs will generate an immediate reply. The video objects are partly cached. Both ES from video object 1 are requested, one of them is not cached. One ES is requested from video object 2, the 2nd never showed up in the SDP and thus, is never requested.

  The ServerSession object simply forwards these requests to the DC objects, which generate SETUP requests and forward them to the server.

- The client receives the responses from the proxy and starts to send PLAY requests for the streams. The synchronization of the ESs is done by the client based on the timing information in the sync layer.

- The ServerSession object parses these requests and invokes the PLAY method at the corresponding DataChannel object.

- The session ends with a TEARDOWN request. The MP4Stream object is updated with the new data and inserted via the cache manager, resulting in a cache replacement algorithm to start.

In the case of a complete cache miss, the scenario changes slightly because the meta-information is needed in advance:

- When the proxy receives the DESCRIBE request, it detects that nothing of the video is cached. A ServerSession object is created, which encapsulates a ClientSession object. The ClientSession object detects that it has to pre-fetch the OD stream and immediately sends a DESCRIBE, followed by a SETUP for the OD and the BIFS stream, the two streams which will always be requested by a client. Afterwards, the OD stream is parsed and an SDP description is generated and sent to the client. This pre-fetching increases the delay for the client, but the delay should be acceptable because both ESs are rather small.

6 Summary and Further Work

In this paper we have presented a design for an adaptive MPEG-4 proxy. We have shown how standards like MPEG-4 and MPEG-7 can be used to realize a standard conform, adaptive proxy. The use of MPEG-7 allows the components in a distributed multimedia system (media server, proxy and client) to communicate adaptation options in a standardized way; MPEG-4 provides the internal media structure to lower adaptation costs in the proxy. The modular design allows for different cache replacement strategies to be integrated and evaluated.

In addition, we support simple filter operations that work in the compressed domain. For future work, we are especially interested in side-effects of the CPU load of the filtering operations on proxy behavior.

References

[1] Rob Koenen. *N4030 - Overview of the MPEG-4 Standard*, available at http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm, March 2001

[2] José M. Martínez. *N4509 - Overview of the MPEG-7 Standard*, available at http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm, Dec. 2001

[3] H. Kosch, L. Böszörmenyi, H. Hellwagner. *Modeling Quality Adaptation Capabilities of Audio-Visual Data*, In DEXA 2001 Workshop Proc., Sep. 2001

[4] Stefan Podlipnig. *Video Caching in Verteilten Multimedia-Systemen*, PhD-Thesis at the University of Klagenfurt/ITEC, April 2002

[5] R. Rejaie and J. Kangasharju. *Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming*, In Proc. NOSSDAV, June 2001

[6] R. Rejaie, H. Yu, M. Handley and D. Estrin. *Multimedia Proxy Caching Mechanisms for Quality Adaptive Streaming Applications in the Internet*, In Proceedings of IEEE Infocom'2000 , Tel-Aviv, Israel, March 2000.

[7] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara. *Proxy Caching Mechanisms with Video Quality Adjustment*, In SPIE International Symposium on The Convergence of Infromation Technologies and Communications, August 2001

[8] M. Zink, J. Schmitt, and R. Steinmetz. *Retransmission Scheduling in Layered Video Caches*, to be published at ICC 2002

[9] L. Böszörményi, H. Hellwagner, H. Kosch, M. Libsie, S. Podlipnig. *Comprehensive Treatment of Adaptation in Distributed Multimedia Systems in the ADMITS Project*, submitted to ACM Multimedia 2002