Architecture of a Quality Based Intelligent Proxy (QBIX) for MPEG-4 Videos*

Peter Schojer, Laszlo Böszörmenyi, Hermann Hellwagner, Bernhard Penz, Stefan Podlipnig
Institute of Information Technology
University Klagenfurt
Klagenfurt, Austria
{pschojer, laszlo, hellwagn, berni, spodlipn}@itec.uni-klu.ac.at

ABSTRACT

Due to the increasing availability and use of digital video data on the Web, video caching will be an important performance factor in the future WWW. We propose an architecture of a video proxy cache that integrates modern multimedia and communication standards. Especially we describe features of the MPEG-4 and MPEG-7 multimedia standards that can be helpful for a video proxy cache.

QBIX supports real-time adaptation in the compressed and in the decompressed domain. It uses adaptation to improve the cache replacement strategies in the proxy, but also to realize media gateway functionality driven by the clients' terminal capabilities.

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications; H.3.4 [**Information Storage and Retrieval**]: Systems and Software; H.5.1 [**Information Systems**]: Multimedia Information Systems

General Terms

Performance, Design

Keywords

Video proxy, video caching, media gateway, media adaptation, MPEG-4, MPEG-7, RTP, RTSP, replacement, LRU

1. INTRODUCTION

Delivering stored video data over the Internet becomes increasingly important for multimedia applications like distance education, digital libraries or video-on-demand systems. During recent years the amount of multimedia objects has increased on the WWW. Especially the number of video objects will increase in the near future and account for a large fraction of data traffic. Videos have three main features that make them different from usual Web objects:

1. They are very large, even compressed.
2. They must be streamed under soft real-time conditions.
3. Their content usually does not change, at least not on a short time scale.

*This project was funded in part by FWF (Fonds zur Förderung der wissenschaftlichen Forschung) P14788 and by KWF (Kärntner Wirtschaftsförderungsfonds).

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM 1-58113-680-3/03/0005.

The first two issues hamper the handling of videos on the Web considerably. The third point, however, is good news, because we can generally ignore the consistency problem in caching of videos. Different techniques have been proposed for appropriate handling of videos in distributed systems. One important technique is the use of *proxies*. Especially interesting are *adaptive, quality aware* proxies, which are able to adapt the actual video content

1. to the given capabilities of the devices/users they serve, acting as a *gateway*;
2. to the given storage capacity of the proxy itself, acting as a *cache*.

As videos are large, simple caching of whole videos will usually perform poorly. Due to the small video-size/cache-size relation, the number of videos will be small in a normal sized cache (few Gigabytes). Increasing the disk capacity will not alleviate this problem as currently used videos are still relatively small (short in time and low in resolution) and will continue to grow in the near future. Simple Web caching techniques proposed for conventional Web objects can only be used if a small number of video objects is responsible for most of the requests. As such strongly skewed popularity distributions are fairly uncommon, the size of video objects introduces indeed a major problem for video proxy caching.

The answer of the QBIX project to this question is the use of adaptive, quality aware caching. Instead of replacing a selected video object in the cache, we reduce its quality (in an integral number of steps) thus trying to save it for later use in reduced quality. We introduce the architecture, implementation and evaluation of such a proxy in this paper. A detailed discussion of quality-aware replacement strategies can be found in [24].

A nice coincidence is that we can use the same adaptation techniques to implement *gateways*. If a proxy can identify classes of usage capabilities, it can serve different classes with different video quality. It obviously makes no sense, e.g., to send high resolution video frames to a low resolution PDA screen, or a colored video to a monochrome screen. A gateway can thus save network bandwidth and client CPU power. The proxy may store different quality versions explicitly, or it may apply quality reduction on the fly.

A special emphasis of the QBIX project lies in relying not only on network-oriented communications standards such as RTP and RTSP, but also on the communications standards of the ISO/IEC Moving Pictures Experts Group (MPEG), designed to transport multimedia data and metadata. This makes the techniques developed generally available and usable.

The organization of the remainder of the paper is as follows: Section 2 describes related work. Section 3 explains adaptation in the context of MPEG-4 and MPEG-7 and explains these two standards

briefly. Section 4 describes the modular design of the proxy and explains each module in detail. Section 5 presents benchmarks on the efficiency of the adaptation algorithms used in the proxy and Section 6 presents a conclusion and further work.

2. RELATED WORK

Web caching has been a very important topic for several years. Especially replacement strategies have attracted a lot of attention. There exist many proposals for replacement strategies; see for example [6, 30]. Furthermore, there exist more specific topics like cache consistency, caching of dynamic content or proxy cache cooperation (hierarchical caching, distributed caching); see [30] for a survey of different Web caching topics.

In this article, we want to concentrate on the basic design of a quality based video caching proxy. One central topic is cache replacement. In contrast to Web cache replacement, video proxy cache replacement has to consider further characteristics of video data (huge amounts of data, possibility of quality adaptation). Due to these characteristics and the growing interest in Web videos, a growing number of video caching strategies has been proposed recently. These strategies can be divided into two categories:

- Full video caching: The whole video is cached at the proxy.
- Partial video caching: A certain part of the video is cached at the proxy.

Full video caching can be found in many available caching products, where videos are handled like typical Web objects. Special commercial solutions for caching streaming videos also cache whole videos[1]. Full video caching can be very resource consuming because videos can be huge compared to conventional Web objects. Therefore, research has been focused on the development of new caching schemes that try to cache only certain parts of the videos near to the clients. We classify these proposals into the following classes:

- Partial caching in the time domain: The cache stores a certain time segment of the video (e.g. beginning part, most important scene). The rest is delivered from the origin server.
- Partial caching in the quality domain: The cache initially stores the whole video but changes the quality and therefore the size of the video according to some criteria.

Examples of partial caching are caching of a prefix [27], prefix and selected frames [18, 16], prefix assisted periodic broadcast of popular videos [10], optimal proxy prefix allocation integrated with server-based reactive transmission (batching, patching, stream-merging) [29], bursty parts of a video [32], hotspot segments [9], popularity-based prefix [23], segment-based prefix caching [31], variable sized chunk based video caching [2] and distributed architectures for partial caching [1, 3] of a video. Some of these caching schemes do not use any dynamic replacement but use periodic cache decisions, e.g. cache replacement is triggered at constant time intervals and not work-load dependent. Some others use simple replacement (LRU) combined with partial caching decisions.

All these proposals are advantageous if reduction of download latency is the main aim or if the potential users behave in a certain way (browse the beginning of videos, prefer certain parts). They are potentially disadvantageous if users try to download whole videos.

[1]Although there is no detailed technical information about these products, they seem to apply this kind of caching.

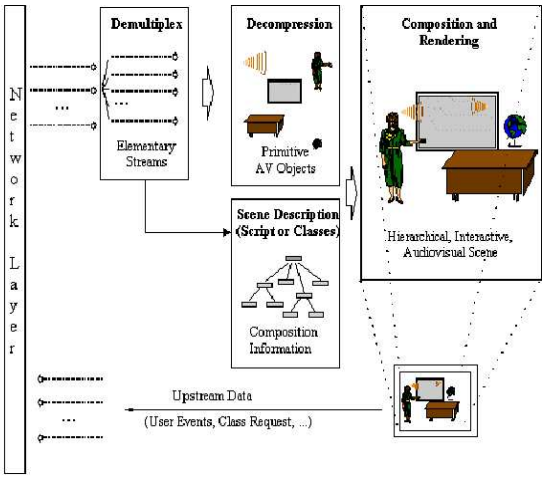


Figure 1: Example of an MPEG-4 Scene [14]

First, there exists a synchronization overhead because the video is distributed over multiple nodes (some parts at the proxy, the rest at the server). Second, interactivity can be a problem. A jump out of the prefix into the suffix can cause some intermission, because the needed data is loaded from the original server.

Other authors propose to cache the whole video but adapt the quality of the videos according to some criteria. Examples are periodic caching of layered coded videos [13], combination of replacement strategies and layered coded videos [22], quality adjusted caching of GoPs (group of pictures) [26], adaptive caching of layered coded videos in combination with congestion control [25] or simple replacement strategies (patterns) for videos consisting of different quality steps [24]. Most of these proposals rely on simulation to evaluate the performance of the caching techniques. Therefore some assumptions have to be made about the structure of the videos (e.g. layered videos).

In this paper, we describe a novel architecture of a video proxy. We decided to use partial caching in the quality domain because of the aforementioned problems of partial caching in the time domain. Furthermore, we implemented a first prototype to evaluate our proposed architecture. This is similar to the implementation described in [25] (the only video proxy implementation we are aware of). But whereas [25] relies on proprietary systems and protocols we try to integrate modern multimedia standards like MPEG-4 and MPEG-7 and modern communication standards like RTP and RTSP. Thus, to the best of our knowledge, this paper introduces the first standard conform implementation of a video proxy/gateway.

3. VIDEO ADAPTATION IN THE CONTEXT OF MPEG-4 AND MPEG-7

In the context of video transmission, adaptation means to transform an already compressed video stream. Media adaptation can be classified into three major categories: bit rate conversion or scaling, resolution conversion, and syntax conversion. Bit rate scaling can adapt to shortages in available bandwidth. Resolution conversion can adapt to bandwidth limitations, but it can also accommodate for known limitations in the user device, like processing power, memory, or display constraints. Syntax conversion is used in a hybrid network to match sender and client compression protocols. While older video coding standards didn't provide extensive support for adaptation, MPEG-4 is actually the first standard that offers exten-

Figure 2: Adaptation on System Level: Object Based Adaptation (from [28])

sive adaptation options.

3.1 MPEG-4

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). MPEG-4 became an International Standard in 1999. Compared to older MPEG standards, it is the first one that offers the possibility to structure audio-visual data. 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 [14].

Each media object consists of at least one *Elementary Stream* (ES). If more than one ES is present for a visual object, the object offers system level adaptation support; see Section 3.2. 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 video 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 [14].

An MPEG-4 movie (with video and 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 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 space. Optionally, one can have several ESs for the video and one for the audio, which adds natural adaptation support to a video [14]. The following paragraphs discuss video adaptation possibilities available in MPEG-4.

3.2 Adaptation in MPEG-4

3.2.1 System Level Adaptation

An MPEG-4 system stream can contain multiple video objects. These video objects may be transmitted with different priorities. Adapting on this level means dropping video objects during transmission (object based scalability). Figure 2 shows an example. Besides object based adaptation, MPEG-4 systems provides spatial, temporal, and SNR fine granular (FGS) scalability support. Combinations of spatial, temporal, FGS, and object-based scalability are possible, although not each combination is allowed (e.g. spatial and FGS). The advantage of adaptation at the system level is that the burden of generating all necessary information for adaptation is in the video production stage. The disadvantage is, however, that possible adaptation options are fixed during encoding and that decoding multi-layer bitstreams adds complexity to the decoder.

3.2.2 Elementary Stream Adaptation

Elementary stream adaptation can be applied on compressed or uncompressed video data. In both cases, adaptation is limited to quality reduction. Adaptation of elementary streams allows for adaptation options not known during the creation of the video in the production stage.

Adaptation on compressed data includes mechanisms for temporal adaptation (frame dropping) and bit rate adaptation (color reduction, low pass filtering, re-quantization [21, 15, 11]). These mechanisms target bit rate adaptation, and can be combined [5]. To a limited extent, format conversions are also possible if both video coding formats share common components which can be reused (e.g. DCT coefficients, motion information) [19, 8].

Finally, adaptation in the pixel domain (on uncompressed video data) is the conventional method for video adaptation. Again, only quality reduction is achievable. Since the video stream is decompressed into raw pixels, which will be encoded again, video adaptation in the pixel domain involves high processing complexity and memory requirements. The advantage of these techniques is flexibility. Video characteristics such as spatial size, color, and bitrate can be modified. Thus, adaptation in the pixel domain can prepare the video according to client properties for optimal resource usage.

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 Adaptation in MPEG-7

MPEG-7 is another ISO/IEC standard developed by MPEG, which became official standard in June 2002. The standard provides a rich set of standardized tools to describe multimedia content. 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*. Descriptors describe audiovisual features of a media stream. Besides descriptions of low-level features of a media stream such as color histograms or shapes, high-level semantic information can also be added. In contrast to video and audio data, meta-data is semi-structured text and thus, easily searchable.

For the proxy, meta-information regarding adaptation is especially important. We restrict ourselves to the content adaptation

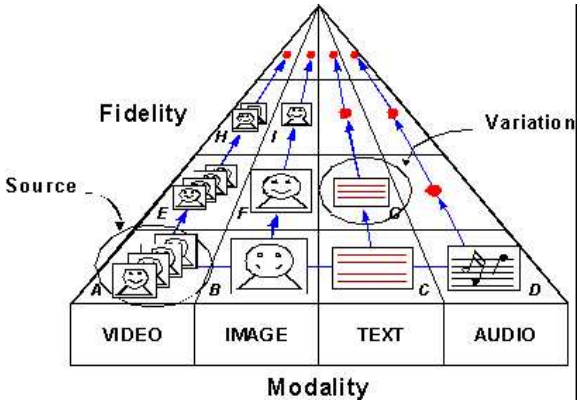


Figure 3: MPEG-7 Variations [12]

part of MPEG-7; for further details see [17, 12].

Figure 3 shows a pyramid of different variations. A video object could be adapted to a video of lower quality, or it could be reduced to one single image. Maybe we have a textual description for this image, so instead of an image, we could send just this textual description, or an audio rendering of this text – depending on the capabilities of the client.

3.4 CC/PP

CC/PP (Composite Capability/Preference Profiles) is a standardized framework developed by the W3C as an extension to the HTTP 1.1 standard. It is a collection of the capabilities and preferences associated with a user and the configuration of hardware, software and applications used by the user to access the World Wide Web. A CC/PP description can be thought of as meta-data of the user's hardware and software. A profile of a client consists of two main blocks. The *Hardware* block describes the resources of the clients, as display size, bandwidth, CPU or memory. The *Software* block stores information on the installed OS and the software capabilities like the supported HTML version or sound support or if images can be viewed.

A small example for a hardware description of a PDA with 16 Mb of memory and a 320x200 display could look like this:

```
<rdf:Description about="HardwarePlatform">
    <prf:Defaults
    Vendor="Nokia"
    Model="666"
    Type="PDA"
    ScreenSize="320x200x16"
    CPU="PPC"
    Keyboard="Yes"
    Memory="16MB"
    Speaker="Yes" />
</rdf:Description>
```

More information on CC/PP can be found at http://www.w3.org/TR/NOTE-CCPP/.

4. ARCHITECTURE

4.1 System Architecture

The proxy cache itself is part of the ADMITS project [4]. The goal of this project is to realize end-to-end adaptive video transport from the media server to the clients as illustrated in the scenario in Figure 4. This scenario includes a multimedia server storing video data, a multimedia database server providing the MPEG-7 information to the media server. In particular, MPEG-7 variation descriptors are being used to control fast adaptation in the network. The video is streamed via several (adaptive) routers to adaptive proxy caches. A proxy cache then sends the data to its clients that can range from low-end mobile devices to powerful PCs. More information on the ADMITS project can be found in [4].

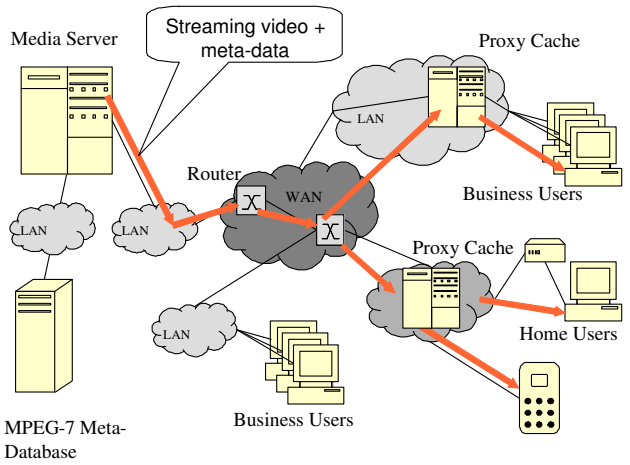


Figure 4: Distributed Multimedia Scenario

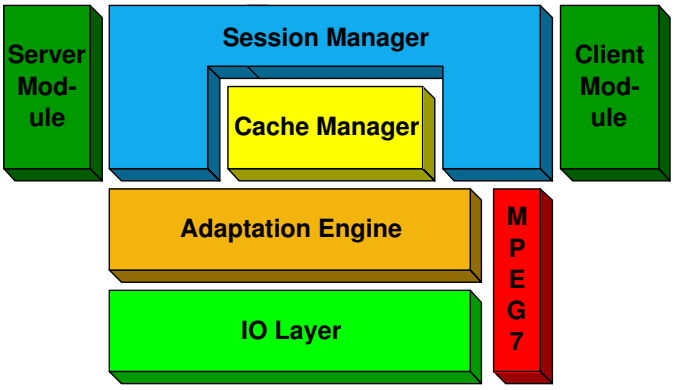


Figure 5: Proxy Modules

4.2 Proxy Architecture

The proxy cache consists of five substantial modules (Figure 5). The *IO Layer* is used to read and write video data, the *Adaptation Engine* uses the IO Layer to read/write frames and transforms them. The *MPEG-7* module offers means to parse and generate MPEG-7 descriptions. The *Cache Manager* manages the cached videos and uses the adaptation engine to realize its cache replacement strategies. The *Session Management* module consists of three modules: The *Server Module* imitates a media server for the client, the *Client Module* imitates a client for the media server and the third is the *Session Manager* that controls the video flow.

4.2.1 IO Layer

The IO layer realizes input/output in the proxy, hiding network and file access behind one abstract class. I/O is frame and Elementary Stream based, i.e., complete frames of an ES are written or read. Currently, raw ES and .mp4 files are supported. On the network, we support multicast and unicast streams packetized with

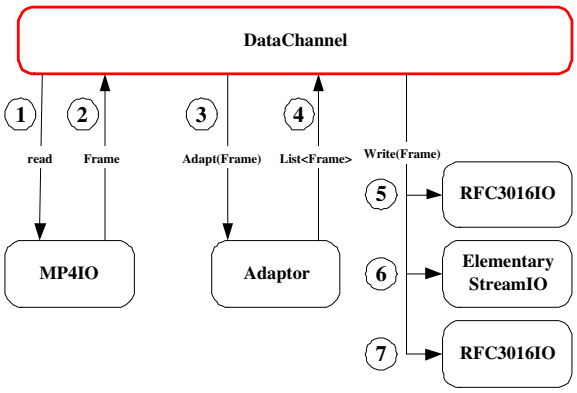


Figure 6: Operation of a DataChannel

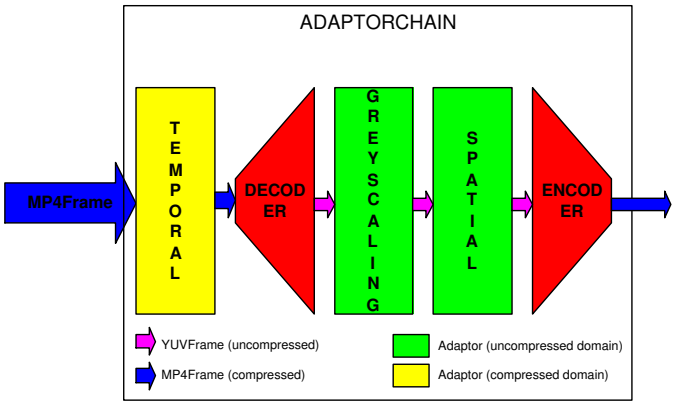


Figure 7: Example of an AdaptorChain

RFC3016[2]. Advanced packetization layers like MultiSL or FlexMux are features currently not supported; they will be added later. For a detailed description of the mentioned packetization layers, see [20].

4.2.2 *Adaptation Engine*

The adaptation engine uses two important concepts:

- Data Channels

  As shown in Figure 6, a DataChannel reads from an IO object (1,2) and invokes an adaptor to (possibly) modify the frame (3). Due to complex adaptors that might require frame buffering, a list of result frames can be returned (4). To cope with such a bursty behavior, the DataChannel maintains a send queue where the result frames are inserted. After the adaptation, only one frame per write-operation is sent to the output IO objects (5,6,7). In the example, we have three different output objects. We have two network destinations (two RFC3016IO objects, single-cast destinations), and one file destination (ElementaryStreamIO, in which case the proxy stores the adaptation result).

  In the worst case, a buffering adaptor will increase the startup delay, but there should be no additional time penalty afterwards (assuming the proxy is fast enough for real-time adaptation).

- Adaptors

  The behavior of an adaptor is quite simple. It expects as input a single frame and returns a list of adapted frames. An adaptor is allowed to buffer frames, until it has enough data available to perform one adaptation step. Currently, only visual media adaptors are supported; system level adaptation is not yet implemented, nor is audio. We support most of the adaptations mentioned in Section 3.2.2. The following ones have been implemented:

  - Temporal Reduction: drop B-frames or B- & P-frames
  - Color Reduction
  - Spatial Reduction
  - Bitrate Scaling

  The TemporalReduction adaptor is implemented as a compressed domain transcoder. It parses the incoming frames, and according to their frame type it decides to drop complete frames or not. To avoid artifacts in the displayed video, frame dropping follows the following rule: first, drop all B frames within a GOP; if this is not sufficient, drop P frames. I frames are not dropped.

  The three remaining adaptors are implemented as pixel domain transcoders. To perform decoding and encoding of an MPEG-4 Video ES, the open-source MPEG-4 codec developed in the XviD project (http://www.xvid.org/) is used. A small wrapper class shields the proxy from the complexity of the decoder and enhances its capabilities to deal with streaming video. The encoder is also wrapped, and only adaptation options are passed to it. Encoder options are: desired bit rate, frame rate, and spatial size of the bitstream. The output picture format of the decoder, which is the same as the input format of the encoder, is set to YV12 colorspace, which is a special case of the YUV colorspace. In YV12, the spatial dimension of the chrominance components U and V is half the size of luminance (Y component) and U and V are swapped leading to the following order: YVU. The encoder behavior is set to constant bit rate (CBR) mode and produces I and P frames only. This reduces the computational complexity significantly, and real-time behavior of the encoder is achieved.

  With the help of the *Decoder* and *Encoder* classes implementing adaptors in the pixel domain is very easy. The BitrateScaling adaptor can be implemented by merely changing the target bit rate of the encoder. For the ColorReduction adaptor two steps are necessary: first, the chrominance components are set to the value gray; second, the bitstream is encoded with reduced bit rate. Tests have shown that chrominance amounts to 20% in the bitstream, therefore a bit rate reduction of 20% is set for the ColorReduction adaptor. The SpatialReduction adaptor is implemented by downsampling the original picture to the desired spatial size. Several algorithms like nearest neighborhood, bilinear, or bicubic interpolation have been developed [7]. For our implementation, we have chosen the simplest one, the nearest neighborhood algorithm. Note that the BitrateScaling, ColorReduction and SpatialReduction adaptors can be implemented in the compressed domain as well to increase performance.

  To allow for maximum flexibility, all adaptors can be arranged in an *AdaptorChain*. Figure 7 shows an example.

[2]http://www.ietf.org/rfc/rfc3016.txt

4.2.3 MPEG-7 Module

The MPEG-7 module adds support for creating and parsing MPEG-7 descriptions. In the current implementation, the focus is on variation descriptors. MPEG-7 describes the internal structure of the source MPEG-4 file and describes size, type (video, audio or BIFS) and bit rate for each ElementaryStream. A variation descriptor contains the name of the adaptation step, the expected quality loss and the priority of this adaptation step. Additionally, a modified MPEG-7 description for each ElementaryStream is generated.

A simplified MPEG-7 description containing an adaptation sequence consisting of two variations might look like this:

```
<Description xsi:type="VariationDescriptionType">
  <VariationSet>
    <Source xsi:type="VideoType"> [..]
      <ComponentMediaProfile id="ES1">  [..]
      <ComponentMediaProfile id="ES2">  [..]
      <MediaFormat>
        <Content href="MPEG7ContentCS">
          <Name>audiovisual</Name>
        </Content>
        <FileFormat
          href="urn:mpeg:mpeg7:cs:FileFormatCS:2001:5">
          <Name xml:lang="en">mp4</Name>
        </FileFormat>
        <FileSize>13107200</FileSize>
        <BitRate>131072</BitRate>
      </MediaFormat>
    [..]
    </Source>
    <Variation id="VARIATION1"
               fidelity="0.75"
               priority="1"> [..]
    <Variation id="VARIATION2"
               fidelity="0.45"
               priority="2"> [..]
  </VariationSet>
</Description>
```

The "source part" describes the internal structure of the MPEG-4 video. For each ES it contains one *ComponentMediaProfile* description, for the complete video it contains one *MediaFormat* block that describes the total size of the video in bytes and the average bit rate of the video.

The following example describes the media profile of one ES of type "visual" with a dimension of 352x288 and a frame rate of 30 frames per second. The size of the video is - as specified in the above example - 12.5 MByte and the bit rate is 128 kBit/sec:

```
<ComponentMediaProfile id="ES1">
  <MediaFormat>
    <Content href="MPEG7ContentCS">
      <Name>visual</Name>
    </Content>
    <VisualCoding>
      <Format
      href="urn:mpeg:mpeg7:cs:... [..]"/>
      <Name xml:lang="en">MPEG-4 Visual</Name>
      <Frame height="288" width="352" rate="30.00"/>
    </VisualCoding>
  </MediaFormat>
</ComponentMediaProfile>
```

The source part is followed by the description of the available variations. A variation description is shown in the next MPEG-7 fragment. The effects of a TemporalReductionAdaptor on the video are described.

```
<Variation id="VARIATION1"
  fidelity="0.75"
  priority="1">
[..]
  <ComponentMediaProfile id="ES1">
   <MediaFormat>
    <Content href="MPEG7ContentCS">
      <Name>visual</Name>
    </Content>
    <VisualCoding>
      <Format href="urn:mpeg:mpeg7:cs:... [..]"/>
      <Name xml:lang="en">MPEG-4 Visual</Name>
      <Frame height="288" width="352" rate="7.50"/>
    </VisualCoding>
   </MediaFormat>
  </ComponentMediaProfile>
  <ComponentMediaProfile id="ES2"> [..]
  <MediaFormat> [..]
    <FileFormat [..]
    <FileSize>6553600</FileSize>
    <BitRate>65536</BitRate>
  </MediaFormat>
[..]
  <VariationRelationship> TemporalReduction
  </VariationRelationship>
</Variation>
```

The proxy knows from this description that applying this adaptor results in a quality loss of 25% (*fidelity="0.75"*), but also that 6.25 MByte are gained and the bit rate is halved for the video.

4.2.4 Cache Management

The Cache Manager (CM) manages all the videos stored in the cache. It uses the adaptation engine to perform the adaptation, and the MPEG-7 module to extract lists of variation sequences (so called *variation sets*) from the MPEG-7 description. In the case no MPEG-7 description is available, a default variation set is used.

The CM actually creates a DataChannel object reading from a source, sending the data through the Adaptor created by the MPEG-7 module, and then saving the output to an ES. As a last optional step, the source ES is deleted.

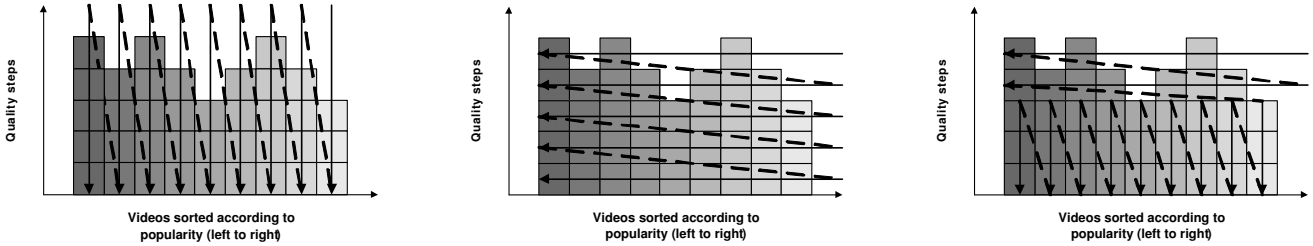
Several cache replacement strategies (CRSs) were integrated into the adaptive proxy. A classical LRU was implemented for comparison reasons, but also advanced CRSs were implemented taking advantage of the adaptation engine and using the meta-information provided by the MPEG-7 module.

Horizontal and vertical cache replacement is supported [24]. The vertical CRS (v-CRS) successively chooses quality variations of the least popular video in the list for replacement. In the worst case, v-CRS degenerates to a classic LRU, especially when many adaptation steps have to be performed to free up enough space disk for one video. As shown in [24], the object hit-rate (hits/requests) is nearly the same when compared to LRU.

The horizontal CRS (h-CRS) chooses the adaptation candidates according to their quality. The video with the highest available quality layer is searched, and adapted. This strategy suffers from a different problem. While h-CRS has a high object hit-rate, its quality hit-rate (average quality of all hits) is low. In the long run, it tends to adapt every object down to its lowest quality layer, even newly inserted videos [24].

To overcome the disadvantages of these two extreme adaptation strategies, a third approach was integrated that combines vertical and horizontal CRSs as illustrated in Figure 8.

The current implementation of CRSs has prototype status. If more than one variation set is present, simply the first set is chosen automatically. A truly intelligent proxy has to compare the two sets and decide which one to use, a feature currently missing but being added in the future. While current LRU strategies are good



(a) Vertical CRS (b) Horizontal CRS (c) Combined CRS

Figure 8: Cache Replacement Strategies [24]

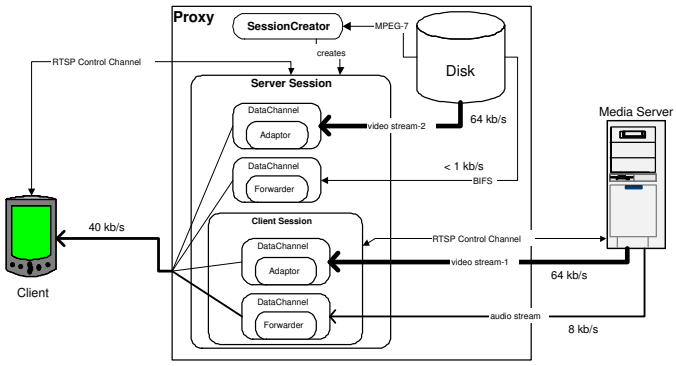


Figure 9: Example of a Session

enough for a first prototype, more advanced CRSs are needed that also consider adaptation costs.

4.2.5 Session Management

The SessionManagement module imitates a server for the client (*ServerModule*) and a client for the media server (*ClientModule*). It is responsible for creating and managing sessions. A *Session* always connects two communication partners. Thus, two different session types exist:

- Server Session: connects client and proxy
- Client Session: connects proxy and server

A Session object manages DataChannels. For each ElementaryStream, one DataChannel is created and controlled. A ServerSession manages all data flows from the local proxy disk, a ClientSession all data flows that read their input from the network.

An exemplary session for a proxy acting as a gateway is shown in Figure 9. The MPEG-4 video in the example consists of four ElementaryStreams (2 video, 1 audio, 1 BIFS, the object descriptor stream is omitted). The client can only accept a video with a bandwidth of 40 kbit/sec. The source video has an original bit rate of 136 kbit/sec. The media server does not offer a lower bandwidth version. Normally, this would exclude the client from viewing that video.

The proxy cache has seen the video previously, thus it has cached the MPEG-7 description, the BIFS stream and one video stream. The second video stream and the audio stream were deleted from the cache and have to be re-fetched from the media server.

Before starting a Session, the client has to communicate its terminal capabilities to the proxy, including display size and the 40 kbit/sec bandwidth limitation. After this step, the client sends an RTSP DESCRIBE to initiate the creation of the Session.

The *SessionCreator* reads the MPEG-7 file from the disk.[3] It then invokes the MPEG-7 module which parses the description and returns a sequence of adaptation steps for this video. The SessionCreator also knows the properties of the video (sent from the media server as an answer to a previous RTSP DESCRIBE) and has detected the mismatch between the bandwidth of the client and the video. This bandwidth gap is closed by setting at each video stream an adaptor. We assume that a BitrateScalingAdaptor is set which reduces the bandwidth from 64 kbit/sec down to 16 kbit/sec for each video stream. The audio stream is simply forwarded. The total bandwidth consumption of the adapted streams is reduced to 40 kbit/sec, so the client can watch at least a lower quality version of the original video. In the extreme case, it could happen that a client has no graphical display and that no video is forwarded at all, only audio.

Note that Figure 9 is simplified. Actually the DataChannels can have more than one client, e.g. in order to both send and store the adaptation result at the proxy. A media gateway with a well defined set of clients could be configured not to store the full quality of the video streams at all but only the adapted versions, whereas a proxy that serves both mobile users and powerful high-end desktop users could store both versions.

5. EVALUATION OF ADAPTORS

The computational performance of the implemented adaptors was tested on different video sequences. The performance measurements presented here are representative for video sequences of the same spatial dimension. Only slight deviations in the range of 1-2 % occurred for different video content.

A representative test video has the following parameters:

- Dimension: 352x288 (CIF)
- Frame rate: 25 fps
- Duration: 23 min 44 sec
- Bitrate: 1 Mbit/sec

The hardware used in the experiments was a 1.6 GHz Pentium-IV system with 512 MB RAM running Windows 2000.

Table 1 and Figure 10 depict where in the adaptation process most CPU resources are spent. Each adaptor chain consists of one

[3]Ideally, there should be no .mp7 file, only one ElementaryStream containing the MPEG-7 information that is part of the MPEG-4 file, but this has not been standardized yet.

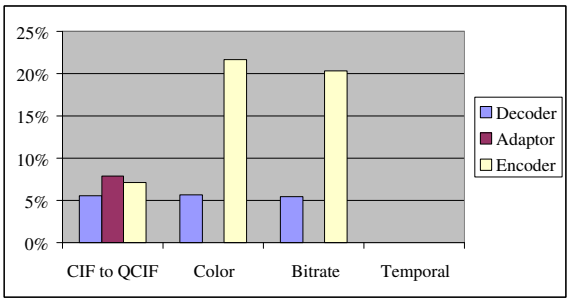


Figure 10: Adaptation Time in % of Total Video Playback Length

Decoder, one Adaptor (SpatialReduction, ColorReduction, or BitrateScaling) and one Encoder, except temporal adaptation where decoding and encoding are not necessary. In the case of BitrateScaling no adaptor is needed, since bitrate scaling is achieved by setting the target bitrate at the Encoder. The CIF to QCIF Adaptor downsamples the video to 176x144.

As expected, temporal adaptation is almost cost-free compared to others. For all adaptors working in the decompressed domain, encoding dominates the transcoding process for large frame sizes (Color, Bitrate). In case of spatial scaling (CIF to QCIF), the number of pixels is reduced by a factor of 4, thus encoding time is rather low.

	CIF to QCIF	Color	Bitrate	Temporal
Decoder	79779	79983	77100	0
Adaptor	113028	281	0	31
Encoder	101405	308183	289759	0
Total	294212	388447	366859	31

Table 1: Time Measurements (in msec) of Different Adaptors Based on a 23 min 44 sec (=1.424.000 msec) Video in CIF Format

6. CONCLUSIONS

We have presented a modular design for an adaptive MPEG-4 proxy. The modular design allows us to build an adaptive proxy cache, a gateway and an adaptive MPEG-4 video server from the same source code base, just with minor modifications.

We have shown how an adaptive proxy overcomes the disadvantages of typical Web proxies. Adaptation improves the object hitrate in the proxy, reducing network load and initial startup delay. The same adaptation algorithms can be used to build a gateway, allowing "poor" clients – normally excluded from viewing the media due to CPU, display or bandwidth constraints – to access the video. Adaptation on videos is an expensive operation. Nevertheless, the evaluation shows that real-time adaptation can be realized with today's hardware, though the number of adaptations executed in parallel is limited. In the worst case, we can allow three clients executing adaptation concurrently, which is acceptable because the number of clients requiring adaptation is – at the moment – assumed to be rather small. This situation will change as soon as mobile devices capable of rendering videos will be widely available and used.

For future work, we have to optimize our adaptation engine and increase the number of adaptation processes allowed in parallel. This can be achieved by integrating new adaptors that work in the compressed domain. Another possibility is system level adaptation, which requires content providers to offer video streams with native scalability options. A third approach is to extend the current single-node proxy towards a distributed proxy architecture or a content distribution network. For efficient adaptation, a combination of all three approaches will be necessary. The proxy itself is still under development, resource management for example is completely missing. Once implementation is finished, an evaluation of the complete system will be done.

7. REFERENCES

[1] S. Acharya and B. Smith. Middleman: A Video Caching Proxy Server. In *Proceedings of the 10th International Workshop on Network and Operating System Support for Digital Audio and Video*, June 2000.

[2] E. Balafoutis, A. Panagakis, N. Laoutaris, and I. Stavrakakis. The Impact of Replacement Granularity on Video Caching. In *Networking*, pages 214–225, May 2002.

[3] E. Bommaiah, K. Guo, M. Hofmann, and S. Paul. Design and Implementation of a Caching System for Streaming Media over the Internet. In *IEEE Real-Time Technology and Applications Symposium (RTAS)*, June 2000.

[4] L. Böszörmenyi, M. Döller, H. Hellwagner, H. Kosch, M. Libsie, and P. Schojer. Comprehensive Treatment of Adaptation in Distributed Multimedia Systems in the ADMITS Project. In *Proceedings of the 10th ACM International Conference on Multimedia*, pages 429–430, Dec. 2002.

[5] C. Kuhmünch and G. Kühne and C. Schremmer and T. Haenselmann. A video-scaling algorithm based on human perception for spatio-temporal stimuli. In *Proc. SPIE Multimedia Computing and Networking (MMCN)*, pages 13–24. SPIE Press, Jan. 2001.

[6] P. Cao and S. Irani. Cost-Aware WWW Proxy Caching Algorithms. In *Proceedings of USENIX Symposium on Internet Technologies and Systems*, pages 193–206, Dec. 1997.

[7] Chuohao Yeo. An Investigation of Methods for Digital Television Format Conversions. Master's thesis, Massachusetts Institute of Technology, May 2002.

[8] E. Amir and S. McCanne and H. Zhang. An application level video gateway. In *Proceedings of ACM Multimedia*, San Francisco, CA, 1995.

[9] H. Fahmi, M. Latif, S. Sedigh-Ali, A. Ghafoor, P. Liu, and L. H. Hsu. Proxy Servers for Scalable Interactive Video Support. *IEEE Computer*, 43(9):54–60, Sept. 2001.

[10] Y. Guo, S. Sen, and D. Towsley. Prefix Caching Assisted Periodic Broadcast for Streaming Popular Videos. In *Proceedings of ICC (International Conference on Communications)*, Apr. 2002.

[11] H. Sun, W. Kwok and J. Zdepski. Architectures for MPEG compressed bitstream scaling. In *IEEE Trans. on Circuits and Systems for VideoTechnology*, volume 6, pages 191–199. IEEE Press, Oct. 1995.

[12] ISO/IEC. FDIS 15938-5 - MPEG-7 Standard - Multimedia Description Schemes. page 539, Oct. 2001.

[13] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross. Distributing Layered Encoded Video through Caches. In *Proceedings of IEEE INFOCOM*, Apr. 2001.

[14] R. Koenen. N4030 - Overview of the MPEG-4 Standard. *mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm*, Mar. 2001.

[15] Z. Lei and N. Georganas. Rate Adaptation Transcoding for Precoded Video Streams. In *Proceedings of the 10th ACM International Conference on Multimedia*, pages 127–136, Dec. 2002.

[16] W.-H. Ma and D. H.-C. Du. Reducing Bandwidth Requirement for Delivering Video over Wide Area Networks with Proxy Server. In *IEEE International Conference on Multimedia and Expo*, pages 991–994, Aug. 2000.

[17] J. M. Martínez. N4509 - Overview of the MPEG-7 Standard. *mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm*, Dec. 2001.

[18] Z. Miao and A. Ortega. Proxy Caching for Efficient Video Services over the Internet. In *9th International Packet Video Workshop (PVW'99)*, Apr. 1999.

[19] N. Feamster and S. Wee. An MPEG-2 to H.263 Transcoder. *SPIE International Symposium on Voice, Video, and Data Communications*, Sept. 1999.

[20] M. Ohlenroth and H. Hellwagner. RTP-Packetization of MPEG-4 Elementary Streams. In *IEEE ICME 2002 International Conference on Multimedia*, volume 2, pages 465–468, Aug. 2002.

[21] P. Assuncao and M. Ghanbari. A frequency-domain video transcoder for dynamic bit rate reduction of MPEG- 2 bit streams. In *IEEE Trans. on Circuits and Systems for VideoTechnology*, volume 8, pages 953–967. IEEE Press, Dec. 1998.

[22] S. Paknikar, M. Kankanhalli, K. R. Ramakrishnan, S. H. Srinivasan, and L. H. Ngoh. A Caching and Streaming Framework for Multimedia. In *Proceedings of ACM Multimedia*, pages 13–20, Nov. 2000.

[23] S. H. Park and K. D. C. E. J. Lim. Popularity-based Partial Caching for VOD Systems using a Proxy Server. In *Workshop on Parallel and Distributed Computing in Image Processing, Video Processing and Multimedia*, Apr. 2001.

[24] S. Podlipnig and L. Böszörmenyi. Replacement Strategies for Quality Based Video Caching. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 2, pages 49–52, Aug. 2002.

[25] R. Rejaie and J. Kangasharju. Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming. In *Proceedings of the International Workshop on Network and Operating Systems Support for Digital Audio and Video*, June 2001.

[26] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara. Proxy Caching Mechanisms With Video Quality Adjustment. In *Proceedings of the SPIE Conference on Internet Multimedia Management Systems*, pages 276–284, Aug. 2001.

[27] S. Sen, J. Rexford, and D. Towsley. Proxy Prefix Caching for Multimedia Streams. In *Proceedings of IEEE INFOCOM'99*, pages 1310–1319, Mar. 1999.

[28] A. Vetro, H. Sun, and Y. Wang. Object-based transcoding for adaptable video content delivery. *IEEE Trans. Circuits and Syst. for Video Tech.*, Mar. 2001.

[29] B. Wang, S. Sen, M. Adler, and D. Towsley. Optimal Proxy Cache Allocation For Efficient Streaming Media Distribution. In *IEEE INFOCOM*, June 2002.

[30] J. Wang. A Survey of Web Caching Schemes for the Internet. *ACM Computer Communication Review*, 29(5):36–46, 1999.

[31] K.-L. Wu, P. S. Yu, and J. L. Wolf. Segment-Based Proxy Caching of Multimedia Streams. In *Proceedings of the Tenth International World Wide Web Conference*, May 2001.

[32] Z.-L. Zhang, Y. Wang, D. H. C. Du, and D. Shu. Video Staging: A Proxy-Server-Based Approach to End-to-End Video Delivery over Wide-Area Networks. *IEEE/ACM Transactions on Networking*, 8(4):429–442, 2000.