Scalable Computing: Practice and Experience
Volume 6, Number 2, pp. 93–104. http://www.scpe.org
ISSN 1895-1767

AN ADAPTIVE STANDARD META-DATA AWARE PROXY CACHE

PETER SCHOJER, LASZLO BÖSZÖRMÉNYI AND HERMANN HELLWAGNER*

Abstract. Multimedia is gaining ever more importance on the Internet. This increases the need for intelligent and efficient video caches. 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 cache 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, the MPEG-7 standard to describe the scalability options for a video and the emerging MPEG-21 standard to describe the terminal capabilities. 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.

Key words. Adaptation, MPEG-4, MPEG-7, MPEG-21, adaptive proxy, caching

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

In this paper we show the design of an adaptive Web proxy using RTSP and the multimedia standards MPEG-4, MPEG-7 and MPEG-21.

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

In [6] an adaptive proxy is introduced. The corresponding implementation is based on a proprietary AT&T client-server architecture, where little public information is available. [8] proposes a proxy that filters GOPs (Group of Pictures) based on the available bandwidth; only simulation results are presented. [9] proposes a way to deal with retransmission of missing segments of layered encoded videos; again simulations are used to derive results. [5] performs an exhaustive comparison of replacement strategies suitable for layered encoded videos, again based on simulation results. [7] 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 [10]. 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 [5] 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 CPU intensive de- and recompression of the video. Fortunately, with MPEG-4 a standard is available that provides support for adaptation.

*University of Klagenfurt, Institute of Information Technology ({pschojer, laszlo, hellwagn}@itec.uni-klu.ac.at)

3.2. Object-based Coding in MPEG-4. MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group). It became an International Standard in the first months of 1999. Compared to older MPEG standards, it is the first one that offers the possibility to structure audio-visual streams. MPEG-4 defines the notion of a scene (see Figure 3.1) 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].

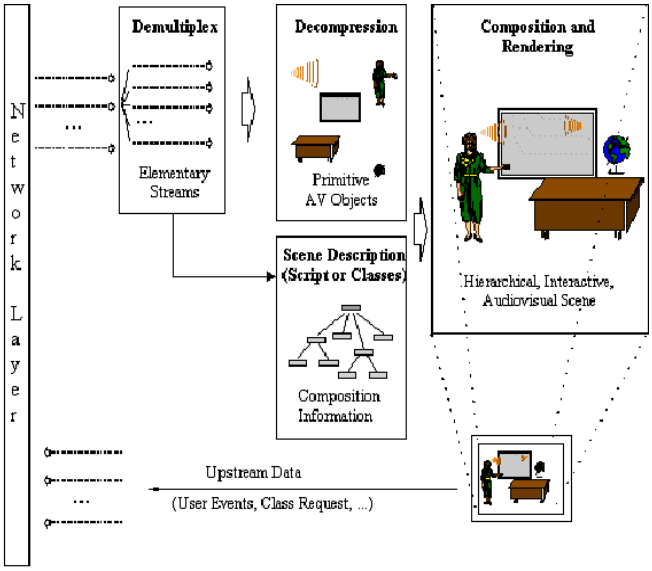


FIG. 3.1. *Example of an MPEG-4 scene [1]*

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 native adaptation support. The additional ESs are used to add spatial, temporal, or SNR scalability to an object. Fast adaptation can be done 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* (AUs), where usually one AU encapsulates one frame. To transfer the AU over the network, the AU is packetized into *SyncLayer packets* and then passed to the delivery layer, where it is packetized into an RTP packet. Depending on the size of the AU, one or more SL packets are needed [1].

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 where exactly it should be rendered. Optionally, one can have several ESs for each object [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. Multimedia Meta-data. 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.

A simplified MPEG-7 description containing an adaptation sequence (=*VariationSet*) 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, i.e., the size is 12.5 MByte and the bit rate is 128 kBit/sec.

The following example describes in detail one ES of type "visual" with a dimension of 352x288 and a frame rate of 30 frames per second:

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

The proxy knows from this description that applying this adaptor will decrease the frame rate down to 7.5 frames, which for this specific video results in a quality loss of only 25% (*fidelity="0.75"*). The proxy also knows that 6.25 MBytes are gained and the bit rate is halved for the video. The priority field is used by the proxy to sort variations within a VariationSet. Unfortunately, the current version of MPEG-7 doesn't support the specification of resource consumptions [4]. But typically, the proxy can estimate how much CPU cycles a variation will use, so this missing feature is not fatal.

3.4. MPEG-21—A Multimedia Framework. MPEG-21 [3] is the latest ISO/IEC development, parts of which will become International Standard in 2004. The goal of MPEG-21 is to define a normative open framework for multimedia delivery and consumption, spanning the whole delivery/consumption chain. We are especially interested in MPEG-21 because it allows clients to declare descriptors for expressing their usage environment. In particular, Part 7—*Digital Item Adaptation* (DIA) [14]—allows specifying terminal capabilities as well as network capability and condition descriptors.

An example of using the terminal capabilities provided by MPEG-21 will be given in Section 6.

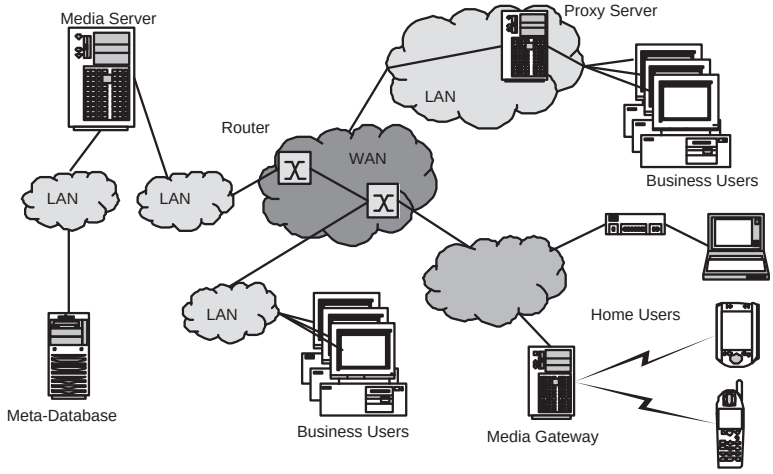


FIG. 3.2. *Components of a distributed multimedia system [10]*

4. The ADMITS Project. The adaptive proxy itself is only one part of an end-to-end adaptation chain. Figure 3.2 shows the layout of a distributed multimedia system. Adaptation options in such a system are investigated in the ADMITS project (*Adaptation in Distributed Multimedia IT Systems*) at our institution [10].

A distributed multimedia system comprises several types of components, such as media servers, meta-databases, proxies, media gateways, routers and clients. Servers, proxies and gateways—being the most powerful components in this scenario—can apply complex adaptation steps, i.e., transcoding. A router can be extended to employ, in case of congestion, a simple packet dropping scheme like dropping B-frames before P- and I-frames. Clients can also employ adaptations like dropping B-frames in case they are unable to decode the full video in real-time, though it is preferable to detect this performance gap at the sender side and not to send the data. In addition to media adaptation, the components of the multimedia system itself can be adapted; one example is server/proxy replication. The main question is, which kind of component can be best used for which type of adaptation.

4.1. Meta-Database. The meta-database supports all kinds of adaptations by supplying meta-data for the adaptation engines. In addition, the meta-database acts for the user as an entry point to the distributed multimedia architecture. It is implemented as an MPEG-7 Multimedia Data Cartridge relying on cartridge technology of the Oracle database management system (DBMS) [15]. The meta-data enables multiple functionalities:

- Support of content-based queries, based on low- and/or high-level indexing information. Low-level indexing favors similarity search, such as, "give me all clips with key frames of similar color distribution as the query image". A high-level query is, e.g., "give me all soccer clips where the Austrian player Vastic scores a goal from a free kick". The meta-database transforms the clients' queries into a sequence of video identifiers and transmits those to the video server, which in turn delivers the videos to the client.
- Support of adaptation based on meta-information, such as the definition of transcoding procedures with the help of MPEG-7 variation descriptors [10].

4.2. Adaptive Router. The operations that a router can perform are fairly limited, since forwarding speed must not be compromised significantly. The advantage is that a router is the best place to cope with varying network load (congestion). The simplest and thus fastest solution is to prioritize frames of a video. In other words, in case of congestion all packets that belong to B-frames are dropped before those belonging to P-frames; I-frames are never discarded, if possible. A new network protocol was developed that provides this prioritization in a way that allows efficient adaptation [16].

4.3. Adaptive Virtual Video Server. The Virtual Video Server (or Adaptive Multimedia Server) is a distributed video server which can change its number of nodes on-the-fly. Videos are stored as a sequence of shots striped over the nodes of the server. For example, if the server detects that a client is far away from the source, it can push the proxy code to a node which resides closer to the client and use this node as a data collector for the requested video [17].

5. Adaptive Proxy.

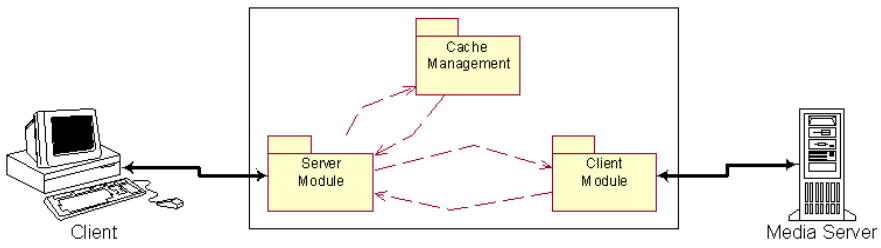


FIG. 5.1. *Proxy architecture*

5.1. Design. 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 5.1 shows how these parts communicate with each other. The server part uses the cache manager to look up a video and to insert videos into the cache. The client part is used to retrieve missing videos/ESs from the server.

5.2. Server Module. The server module listens for RTSP requests and manages existing sessions. For a new request a *ServerSession* object is created. 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 ClientSession 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 5.2. To keep the example concise, the data flows writing the data to the disk are omitted.

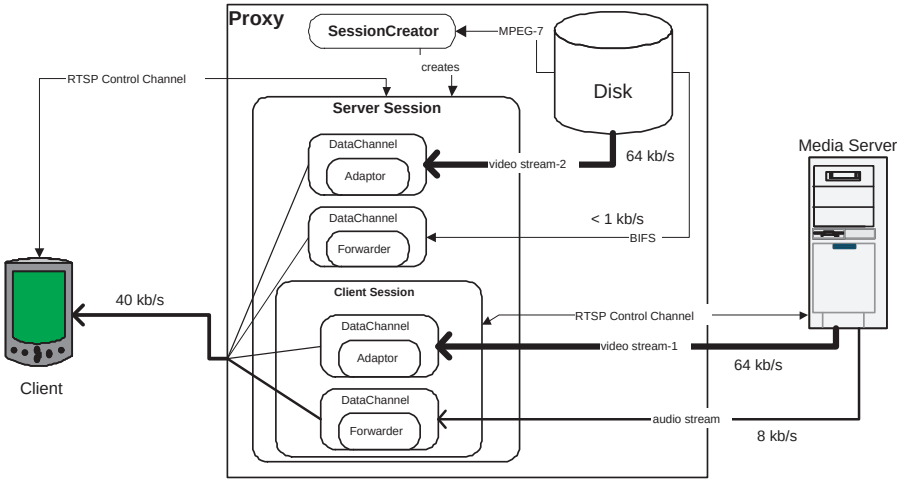


FIG. 5.2. *Data flow with adaptation*

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

- System level adaptation
  An MPEG-4 video can consist of several video objects. This allows for object based scalability (dropping video objects) as shown in Figure 5.3. Furthermore, each object can consist of several ESs. Having more than one ES per video object adds native support for *temporal*, *spatial* or *SNR* scalability—if the media supports it and has been encoded by the content provider to offer enhancement layers or objects. The decision to apply system level adaptation 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).
- 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)
  - Color reduction (grey-scaling)
  - Spatial reduction
  - Bit rate reduction.

  The temporal scaling adaptors work in the compressed domain. All other operations require CPU intensive de- and recompression. More information on the adaptation process and benchmarks on the CPU load generated by one adaptation step can be found in [11].

5.3.1. 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.

FIG. 5.3. *Adaptation on system level: object based adaptation (from [12])*

This information can be extracted from the OD stream and is needed rather early during the RTSP DESCRIBE phase. In 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 meets 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 [10].

5.4. 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 sequence

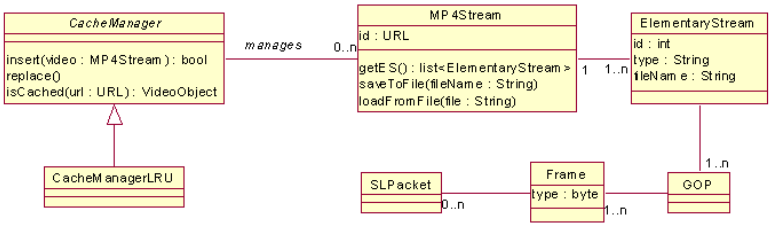


FIG. 5.4. *Cache management*

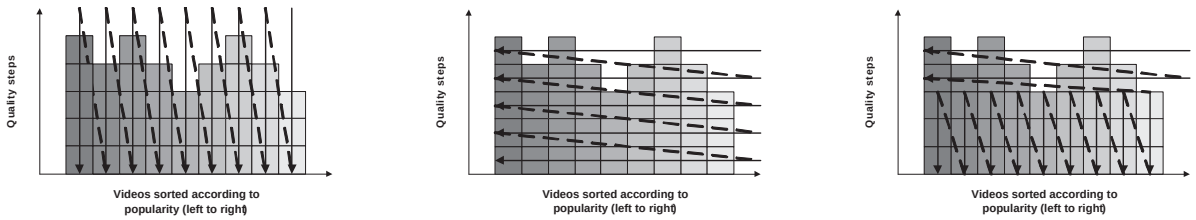
of GOPs, which consist of frames, which are stored as a sequence of SyncLayer [1] packets (Figure 5.4). The *CacheManager* class itself is defined as an abstract class, allowing several cache replacement strategies to be realized.

5.4.1. Cache Replacement. Several cache replacement strategies (CRSs) are offered by the proxy. A classical LRU was implemented for comparison reasons and more advanced variants of LRU that allow lowering the quality of a cached video in discrete steps.

Horizontal and vertical cache replacement is supported [13]. 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 [13], 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 [13].

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 5.5.



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

FIG. 5.5. *Cache replacement strategies [13]*

6. Example. Figure 6.1 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. In Figure 6.1 *SDP–* or *data–* means that an adapted version is sent to the client. We will explain a partial cache hit in detail.

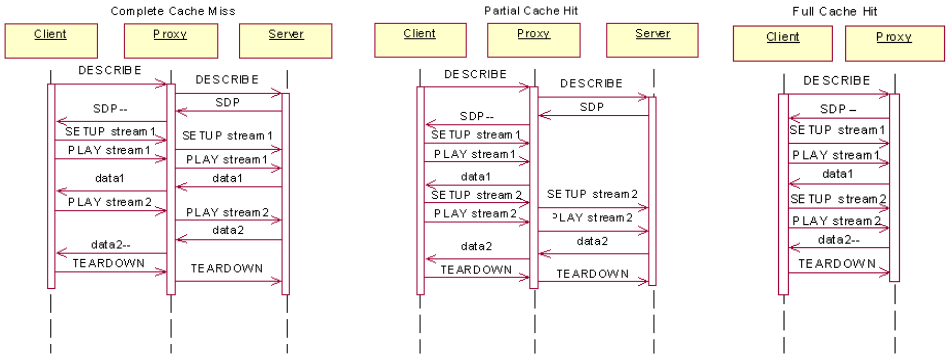


FIG. 6.1. *Caching scenarios*

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. For a first prototype, we have appended this information as an MPEG-21 *Digital Item* [14] directly to an RTSP DESCRIBE command. In the following example, a PDA with an 8-bit grey-scale display, a resolution of 240x320 and a network capability up to 128 kBit/sec, requests a video. The device supports mono audio.

```
DESCRIBE rtsp://127.0.0.1/videoandaudio.mp4 RTSP/1.0
CSeq: 1
Accept: application/sdp
User-Agent: MPEG4ITEC Player
Content-Type: application/mpeg21_dia
Content-Length: 458
<xml version="1.0" encoding="UTF-8"?>
<DIA xmlns="urn:mpeg:mpeg21:dia:schema:2003"
  xmlns:mpeg7="urn:mpeg:mpeg7:schema:2001"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="urn:mpeg:mpeg21:dia:schema:2003">
<Description xsi:type="UsageEnvironmentType">
  <Terminal>
    <InputOutput>
      <Display bitsPerPixel="8" colorCapable="false">
        <Resolution horizontal="240" vertical="320"/>
      </Display>
      <AudioOut numChannels="1"/>
```

```
    </InputOutput>
    <DeviceProperty>
      <Storage size="64.0" writable="false"/>
      <DeviceClass>PDA</DeviceClass>
    </DeviceProperty>
  </Terminal>
  <Network>
    <Capability maxCapacity="128000" minGuaranteed="32000"
          inSequenceDelivery="false" errorDelivery="true"/>
  </Network>
</Description>
</DIA>
```

- 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* (DCs); 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 media server.
  If no MPEG-7 information is cached, the proxy fetches an MPEG-7 description from the media server. This description is used by cache replacement strategies in the proxy but could be used by a content provider to restrict or forbid adaptation. For instance, imagine a commercial video server environment where users pay a certain amount of money for watching a full-quality video; in this case the proxy should not modify the video in any manner.
  If the terminal capabilities indicate that the end-device cannot 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 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 desired number of ESs. An example for such a reply including IOD and one video ES description is:

```
RTSP/1.0 200 OK
Server: ItecMp4Server
CSeq: 1
Content-Type: application/sdp
Content-Length: 736
Date: 31 Jan 2003 11:52:22 GMT
Expires: 31 Jan 2003 11:52:22 GMT
Content-Base: rtsp://127.0.0.1/videoandaudio.mp4/

v=0
o=StreamingServer 1044010342 840 IN IP4 143.205.122.43
c=IN IP4 143.205.122.43
b=AS:112
t=0 0
a=control:*
a=mpeg4-iod:"data:application/mpeg4-iod;base64,AoCAgGOAT [..]"
a=isma-compliance:1,1.0,1
a=range:npt=0-13.23300
m=video 0 RTP/AVP 96
b=AS:112
a=rtpmap:96 MP4V-ES/90000
a=control:trackID=1
a=cliprect:0,0,240,180
```

```
a=fmtp:96 profile-level-id=1;config=000001000000012000044006685821 20A31F
a=mpeg4-esid:1
```

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

```
SETUP rtsp://127.0.0.1/videoandaudio.mp4/trackID=1 RTSP/1.0
CSeq: 2
Transport: RTP/UDP;unicast;client_port=40002-40003;
```

- The proxy sees 6 SETUP requests, which are forwarded to their corresponding DCs. Assuming that the OD, BIFS, and audio streams are cached, their DCs will generate immediate replies. The video objects are partly cached. Both ESs from video object 1 are requested, one of them is not cached. One ES is requested from video object 2, the second never showed up in the SDP and thus, is never requested. The ServerSession object simply forwards requests to non-cached streams to the ClientSession, which generates SETUP requests and forwards them to the media server.

- The client receives for each SETUP a response from the proxy:

```
RTSP/1.0 200 OK
CSeq: 2
Server: ItecMp4Server
Last-Modified: 31 Jan 2003 11:52:22 GMT
Cache-Control: must-revalidate
Session: 8162240
Date: 31 Jan 2003 11:52:22 GMT
Expires: 31 Jan 2003 11:52:22 GMT
Transport: RTP/UDP;unicast;client_port=40002-40003;source=127.0.0.1;
server_port=20000-20001
```

  It extracts the SessionId and the ports. Now the client has all the information necessary to create an RTP connection to 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.

```
PLAY rtsp://127.0.0.1/videoandaudio.mp4 RTSP/1.0
CSeq: 8
Session: 8162240
Range: 0.0 - 13.233000
```

- 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 (CR) algorithm to start. The CR algorithm chooses a video to adapt, it analyzes the adaptation possibilities extracted from the MPEG-7 document, and chooses from several adaptation sets the one that offers the best quality in relation to the resulting file size.

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.

7. Summary and Further Work. We have presented the design of an adaptive MPEG-4 proxy. We have shown how standards like MPEG-4, MPEG-7, and MPEG-21 can be used to realize a standard conforming, adaptive proxy. The use of MPEG-7 and MPEG-21 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 and uncompressed domains that allow our proxy to act as a media gateway and to tailor a video to a client's terminal capabilities. While the implementation of the aforementioned filters is completed and first evaluations have been performed [11], the integration of system-level adaptation turned out to be a hard task. There are no free codecs available that allow one to create videos with native scalability support. As soon as tool and codec support is improving, we will integrate this adaptation possibility into our proxy. Another open issue is the side effect of the CPU intensive filtering operations on proxy behavior. If we want to maintain real-time behavior in the proxy, we are limited to a few adaptation steps in parallel. There are several possibilities to cope with this constraint. The first and simplest is to declare that the situation will improve with the availability of more advanced codecs. The second variant is to integrate optimized transcoding algorithms in our proxy that work in the compressed domain. Last but not least, extending the current single-node solution to a distributed proxy architecture is also an option.

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

REFERENCES

[1] R. Koenen (ed.), *N4668 - Overview of the MPEG-4 Standard*, available at `http://mpeg.telecomitalialab.com/standards/mpeg-4/mpeg-4.htm`, March 2002
[2] J. M. Martínez (ed.), *N4980 - MPEG-7 Overview*, available at `http://mpeg.telecomitalialab.com/standards/mpeg-7/mpeg-7.htm`, July 2002
[3] J. Bormans and K. Hill (eds.), *N5231 - MPEG-21 Overview*, available at `http://mpeg.telecomitalialab.com/standards/mpeg-21/mpeg-21.htm`, Oct. 2002
[4] H. Kosch, L. Böszörmenyi, and H. Hellwagner, *Modeling Quality Adaptation Capabilities of Audio-Visual Data*, In DEXA 2001 Workshop Proc., Sept. 2001
[5] S. Podlipnig, *Video Caching in Verteilten Multimedia-Systemen*, Ph.D. Thesis, University of Klagenfurt/ITEC, April 2002
[6] R. Rejaie and J. Kangasharju, *Mocha: A Quality Adaptive Multimedia Proxy Cache for Internet Streaming*, In Proc. NOSSDAV, June 2001
[7] R. Rejaie, H. Yu, M. Handley, and D. Estrin, *Multimedia Proxy Caching Mechanisms for Quality Adaptive Streaming Applications in the Internet*, In Proc. IEEE Infocom'2000, Tel Aviv, Israel, March 2000
[8] M. Sasabe, N. Wakamiya, M. Murata, and H. Miyahara, *Proxy Caching Mechanisms with Video Quality Adjustment*, In Proc. SPIE International Symposium on The Convergence of Information Technologies and Communications, August 2001
[9] M. Zink, J. Schmitt, and R. Steinmetz, *Retransmission Scheduling in Layered Video Caches*, In Proc. ICC 2002, New York, April 2002
[10] L. Böszörményi, H. Hellwagner, H. Kosch, M. Libsie, and P. Schojer, *Comprehensive Treatment of Adaptation in Distributed Multimedia Systems in the ADMITS Project*, In Proc. ACM Multimedia 2002 Conference, Juan Les Pins, France, Nov./Dec. 2002
[11] P. Schojer, L. Böszörményi, H. Hellwagner, B. Penz, and S. Podlipnig, *Architecture of a Quality Based Intelligent Proxy (QBIX) for MPEG-4 Videos*, In Proc. World Wide Web Conference 2003, Budapest, Hungary, May 2003
[12] A. Vetro, H. Sun, and Y. Wang, *Object-based transcoding for adaptable video content delivery*, In IEEE Trans. Circuits and Syst. for Video Technology, March 2001
[13] S. Podlipnig and L. Böszörmenyi, *Replacement Strategies for Quality Based Video Caching*, In Proc. IEEE Int'l. Conf. on Multimedia and Expo (ICME), Lausanne, Switzerland, Aug. 2002

[14] Moving Picture Experts Group, *ISO/IEC JTC 1/SC 29/WG 11 N5353: Text of ISO/IEC 21000-7 CD - Part 7: Digital Item Adaptation*, available at `http://www.itscj.ipsj.or.jp/sc29/open/29view/29n5204c.htm`, Dec. 2002
[15] M. Döller and H. Kosch, *Demonstration of an MPEG7 Multimedia Data Cartridge*, In Proc. ACM Multimedia 2002 Conference, Juan Les Pins, France, Nov./Dec. 2002
[16] M. Ohlenroth and H. Hellwagner, *A Protocol for Adaptation-Aware Multimedia Streaming*, In Proc. IEEE Int'l. Conf. on Multimedia and Expo (ICME), Baltimore, USA, July 2003
[17] R. Tusch, *Towards an Adaptive Distributed Multimedia Streaming Server Architecture Based on Service-oriented Components*, Joint Modular Languages Conference, Klagenfurt, Austria, August 2003

Edited by: Zsolt Nemeth, Dieter Kranzlmuller, Peter Kacsuk, Jens Volkert
Received: March 21, 2003
Accepted: June 5, 2003