Metadata driven adaptation in the ADMITS project

László Böszörményi, Hermann Hellwagner*, Harald Kosch, Mulugeta Libsie, Stefan Podlipnig
Department of Information Technology, University Klagenfurt, Universitatsstrasse 65-67, Klagenfurt A-9020, Austria

Abstract

The ADMITS project (Adaptation in Distributed Multimedia IT Systems) is building an experimental distributed multimedia system for investigations into adaptation, which we consider an increasingly important tool for multimedia systems. A number of possible adaptation entities (server, proxy, clients, routers) are being explored, different algorithms for media, component and application-level adaptations are being implemented and evaluated, and experimental data are being derived to gain insight into when, where and how to adapt, and how individual, distributed adaptation steps interoperate and interact with each other.

In this paper the “adaptation-chain” of (MPEG-conforming) metadata based adaptation is described: from the creation stage at the server side, through its usage in the network (actually in a proxy), up to the consumption at the client. The metadata are used to steer the adaptation processes. MPEG-conformant metadata, the so-called variation descriptions, are introduced; an example of a complete MPEG-7 document describing temporal scaling of an MPEG-4 video is given. The meta-database designed to store the metadata is briefly discussed. We describe how the metadata can be extracted from MPEG-4 visual elementary streams and initial results from a temporal video scaling experiment are given. We further present how the metadata can be utilized by enhanced cache replacement algorithms in a proxy server in order to realize quality-based caching; experimental results using these algorithms are also given. Finally, an adaptive query and presentation interface to the meta- and media database is outlined.

Keywords: Adaptation; Metadata; Multimedia Database; Video Caching; MPEG-4; MPEG-7; MPEG-21

1. Background and rationale

Adaptation is becoming an increasingly important tool for resource and media management and delivery in distributed multimedia systems. Best-effort scheduling and worst-case reservation of resources are two extreme cases, none of them well suited to cope with large-scale and dynamic multimedia systems. The middle course can be taken by having a system which dynamically adapts its data, resource requirements, and processing components to achieve user satisfaction. Nevertheless, there is no agreement as to where, when and what to adapt and who should adapt. A number of papers have been published in recent years, where adaptation is a central issue, mostly in different interpretations and generally...
in a somehow limited scope; examples are [2,6,18–20,24].

A distributed multimedia system comprises several types of components, such as media servers, meta-databases, proxies, routers, clients (Fig. 1). Also, a large number of adaptation possibilities exist, from simple frame dropping up to virtual server systems which dynamically allocate new resources on demand. The main question that needs to be addressed is then: which component can be best used and for what kind of adaptation. In the ADMITS project, we are seeking for answers to exactly this basic question, and to a number of related questions.

The problem space is demarcated by the following taxonomy:

1. What is adaptation? Adaptation is the capability of a system to dynamically change its behavior in order to keep the quality of service (QoS) above a certain level. A simple example is a router dropping B-frames of an MPEG video if the available bandwidth degrades. A more sophisticated example is a proxy cache that reduces the quality of replacement candidates (videos) instead of discarding them, thus keeping the startup-delay for video playout low. Note that QoS is a complex issue in itself—adaptation typically reduces some quality parameters in order to keep the overall multimedia experience at an acceptable level.

2. Global versus local adaptation: Adaptation is often considered in a narrow, local context, regardless of the positive or negative interfering effects of individual adaptation measures. However, in a large distributed system, it makes sense to take a global view on different levels and places of adaptation.

3. Adaptation versus reduction: We often speak about adaptation when we actually mean quality reduction. Adaptation includes, however, also dynamic improvement of quality.

4. Proactive versus reactive adaptation: Reactive adaptation is triggered by changes in the settings of a given environment. Proactive adaptation is based on the anticipation of such changes. Recent standardization activities—especially MPEG-4 [11], MPEG-7 [16] and MPEG-21 [13]—favor the provision of metadata, which enable both to signal and effectively perform relevant changes.

5. Types of adaptations: Degrees of freedom in adaptation exist in at least four major dimensions:
   (a) Available resources.
   (b) Representation of the audio/visual (A/V) data.
   (c) Needs of the applications.
   (d) Places of adaptations.

All the above dimensions are discrete, and usually differently restricted for a given environment. In one case, we may have a lot of freedom in changing the representation of the A/V data, but no way to get additional resources; in another setting, allocation of new resources on demand may be easier than transcoding of the media data. The different kinds of adaptations differ—besides their effectiveness and efficiency—mainly in the answer to the question: which of these dimensions are regarded as fixed and which ones may change dynamically? Also, not all combinations are meaningful for practical usage.

We introduce the following classification:

1. Media adaptation: We select a suitable representation depending on the given needs of the application and on the available resources. We may, for example, drop enhancement layers in a layered video-coding scheme if bandwidth is scarce or we may reduce spatial resolution if the...
presentation device has a small-resolution display anyway.

2. **Component adaptation:** We change (usually improve) the characteristics of the given components. We may, for example, load an additional decoding algorithm, or we may allocate a number of additional computing nodes, communication channels, etc. In such kind of adaptation, media representation and application remain unchanged. This kind of adaptation could also be called *offensive* as it tries to allocate additional resources, as opposed to the more usual *defensive* strategy where we try to make use of the best out of the available.

3. **Application-level adaptation:** Given a certain amount of resources, we change (typically reduce) the needs of the application, without violating the real needs of the users. We may, for instance, use a smaller window size, or present only the audio data and refrain from presenting the video data, etc. The crucial point is that the application behaves in a cooperative way instead of greedily insisting on the maximum QoS level.

All three kinds of the above adaptations can be done in any of the components of the distributed environment of Fig. 1; but this does not mean that they should be done everywhere.

Moreover, the individual adaptation dimensions may themselves span a multidimensional space, such as the *InfoPyramid*, which provides the basis for a complex value-resource-based optimum search in the $A/V$ representation dimension, as described in [19]. Algorithms are proposed to implement optimal selection of transcoding algorithms, e.g., in [19,22]. However, for the complex, at least four-dimensional space as described above, we have no chance for effective and efficient solutions without practically relevant heuristics. Therefore, it is not sufficient to regard the above issues as abstract optimization problems, rather experimental data must be gathered to enable good heuristics.

In the ADMITS project, an experimental system comprising all possible adaptation points is being built, and a number of algorithms for media, component and application-level adaptations are being implemented and evaluated. As a main result, we are getting experimental data and particular insights that help to find the proper place for the proper kind of adaptation. In other words, we aim at global adaptation based on properly distributed individual adaptation measures. This is in accordance with newer approaches in peer-to-peer networking, such as the *Media Accelerating Peer Services (MAPS)* as described in [15]. This approach pays a lot of attention to address heterogeneity and scalability, which are also the central goals in using adaptable systems.

In the sense of addressing the multiple facets of adaptation as introduced above, the ADMITS project goes beyond the widely known *Universal Multimedia Access (UMA)* framework, as pursued, for example, by projects at EPFL, Lausanne [http://itswww.epfl.ch/~newuma] and NTNU, Trondheim [http://www.midgardmedia.net/uma.htm] [21], Columbia University, New York [http://www.ctr.columbia.edu/~ywang/Research/UMA/], or Siemens, Munich (Multimedia Message Box) [9]. However, these UMA projects provide important results and insights into media adaptation and associated auxiliary tools such as terminal and network capability descriptions as well as user preference descriptions. The latter also holds for the current activities in MPEG-21 Digital Item Adaptation [13]. The results of these efforts provide important input to the ADMITS project regarding the media adaptation part of the work.

The remaining part of this paper is organized as follows. A short overview of the ADMITS project and its components is given in Section 2. We will discuss the type of metadata that is required for the adaptation process and its representation in a meta-database in Section 3. Section 4 is devoted to the functionality and architecture of the metadata generation component, called the Processing Unit, and also presents experimental results of metadata generation. We present how the metadata is used on the delivery path to the user (actually in a proxy cache) in Section 5. The client side is discussed in Section 6. Finally, Section 7 summarizes the paper and gives an insight into the future.
2. The ADMITS project

In this Section we present the architecture realized in ADMITS. We will describe the components of our solution from metadata creation to consumption. These components are presented later in detail in their respective sections. Section 2.1 discusses the need for standardized media, metadata, and communication. Subsequent sections discuss the adaptive server, proxy cache, and query and presentation interface.

2.1. Reliance on standards and metadata

Efficient adaptation in the sense discussed in Section 1 requires that the participating components know each other and take advantage of adaptation steps done by other components. This requires standardized media, metadata, and communication. Fortunately, recent standardization activities in MPEG go exactly in this direction. MPEG-7 defines the Variation Description Scheme, which enables standardized communication of A/V data in different representations [16]. As a consequence, all components of the ADMITS project “speak and understand” MPEG-7; MPEG-7 metadata serve as the common denominator. More recently, MPEG-21 has started activities on Digital Item Adaptation (DIA), which enables standard communication of dynamic adaptation of both media resources and metadata [13]. MPEG-21 is, however, out of the scope of this paper.

The ADMITS project realizes an architecture shown in Fig. 2 which is similar to that in Fig. 1. It gives a walkthrough of the metadata driven adaptation in the end-to-end multimedia scenario. The numbers in the shaded circles indicate the sequence of events in the adaptation process. The individual components play different roles in the adaptation process. They are connected physically by the network and “semantically” by MPEG-7 metadata that flow over the network in conjunction with the media data. The major components together with the metadata life-cycle are discussed in the sequel.

2.2. Meta-database

The meta-database supports all kinds of adaptations by supplying metadata. Queries formulated by the clients are forwarded to the meta-database, containing data about the A/V data stored on the media server. The metadata enables multiple functionality:

1. Support of content-based queries, based on low- and/or high-level indexing information.
2. Support of adaptation based on meta-information, such as the definition of transcoding procedures with the help of MPEG-7 Variation Descriptors.

2.3. Adaptive virtual video server

The adaptive virtual video server [8] provides means for offensive component adaptation. The video server stores the raw A/V data. It has a distributed architecture containing a number of nodes. In particular, it is composed of a media storage server and a meta-database (Fig. 2). Content indexing for obtaining relevant metadata is done in two ways: the media is analyzed for semantic and structural content (Step 1a) and for its adaptation capabilities (Step 1b). Therefore, we extract two categories of metadata in two steps:

- **Step 1a:** In this step, video segmentation is carried out and semantically meaningful information is extracted, e.g., which events, persons, and objects are the important entities of a segment. Furthermore, low-level descriptors are extracted, for instance, the MPEG-7 DominantColor and the ScalableColorType. MPEG-7 Semantic and structural Descriptors enable content-based queries. More information on storage and annotation of MPEG-7 is found in [4].
- **Step 1b:** In this step, MPEG-7 Variation Descriptors are produced. These descriptors express the relationship between the available variations of the media and their characteristics in terms of media information (e.g., file size and frame rate) and quality.
The ADMITS server provides delivery functionality for media and its descriptive metadata. Therefore, upon a media request, the media server contacts the meta-database for descriptions (Step 2a). These descriptions are attached to the media stream and delivered to the client (Step 2b).

The server is called “virtual” because it is able to change the set of actually allocated physical nodes on demand. It relies on a mobile agent-based infrastructure [7], conceptually closely related to the virtual server system described in [5]. The infrastructure recommends nodes to be allocated in order to implement offensive adaptation. For example, in a specific scenario, a component called the Host Recommender might notice that a client is “far” from the nodes storing the stripe units (e.g., due to a slow connection) and, therefore, it would be desirable to allocate a new node near to the client to act as a data collector and streaming proxy. The proxy functionality is loaded by the so-called Application Loader. After that, the other nodes of the server push their stripe units to the new node, which collects and streams them to the client.

2.4. Adaptive proxy cache

This component supports media adaptation. The proxy cache implements improved quality-aware versions of the well-known LRU and greedy dual-size algorithms. (For a detailed study of the algorithms see [23].) The cache initially stores full videos with their respective metadata descriptions in the form of MPEG-7 Variation Descriptors, and it reduces their quality (and thus their size) in integral steps, before fully deleting them. Fortunately, the relation between size and quality reduction is usually non-linear up to a certain limit, and hence large size reduction usually causes a moderate quality loss. Thus, the cache can offer both short start-up delay and acceptable
quality. The essential new idea is that the quality reduction process is driven by metadata in a standardized format (MPEG-7), which is readily provided in ADMITS by the meta-database (Step 3 of Fig. 2). For each adaptable video, the cache expects just the simple triple parameter: transcoding operation, resulting size, resulting quality.

2.5. Adaptive routers

Routers with enriched functionality may support media adaptation as well. In contrast to a proxy cache, the operations that a router can perform on a media stream are fairly limited, since the forwarding speed must not be compromised significantly. On the other hand, a router is best positioned, for example, to cope up judiciously with dynamically varying network load (most importantly, congestion) or to multicast media streams along heterogeneous links or toward clients with different capabilities. Typical adaptation operations of routers would be dropping of video frames, of certain enhancement layers, or of entire MPEG-4 Elementary Streams. The challenge in this context is to provide and efficiently encode and communicate metadata for routers that convey this information and enable them to perform effective media adaptations. Adaptive routers will not be considered further in this paper.

2.6. Adaptive query and presentation interface

The adaptive query and presentation interface supports the specification of search criteria for media resources, displays results from the database (metadata), proposes means for selecting the media from the database results, and opens players for the selected media. It is the final adaptation level in the end-to-end scenario as depicted in Fig. 2, Step 4. It is adaptive in the sense that all interface components adjust dynamically to the usage environment. The usage environment comprises the client’s terminal capabilities (e.g., hardware, software) and the usage preferences (e.g., user prefers only audio files).

3. Metadata for adaptation

In this Section we discuss the type of metadata that are required in the adaptation process and their representation in a meta-database. Section 3.1 presents the MPEG-7 descriptors that are required for adaptation. In Section 3.2 we propose the VariationSetTree which will allow us to represent the descriptions for a chain of adaptations. Finally Section 3.3 describes the meta-database that serves as a repository of all adaptation and query related metadata.

3.1. Variations and VariationSets

Interoperable and transparent access to multimedia resources, on different types of terminals and/or over diverse networks, is one of the key goals of adaptation and the major theme of ADMITS. Towards this goal and in the context of MPEG-7 [16], a variety of Descriptors have been proposed that can support multimedia resource adaptation in network nodes on the delivery path to the client. In particular, the VariationSet Description Scheme (DS) is useful to the adaptation process by providing hints as to when to apply an adaptation and which algorithm to use. The VariationSet DS may be understood as holding an unbounded set of variations of a multimedia resource described by a Variation DS. It allows one to bind several versions of the same resource without specifying how these versions are obtained. For instance, if the source resource is a video, then the variations may consist of an image, audio, or several videos with lower temporal, color or spatial resolutions obtained respectively by temporal, color or spatial reduction.

The two key concepts used in the VariationSet DS are:

1. Variation: The Variation includes information specifying the Fidelity, Priority, and the Relationship type of a variation multimedia resource. The Fidelity attribute indicates the fidelity of the variation content with respect to the source content, i.e., it expresses how “close” or “faithful” the variation is to the original. The Priority attribute indicates the priority of the
variation resource with respect to the other Variations that may be included in a VariationSet. The Variation's Relationship may be chosen from one of 18 predefined types, like summarization, various kinds of reductions, scaling translations, etc., or a reference to an external classification schema.

2. Source and Variation Profiles: The VariationSet describes the source and variation resources using the MultimediaContent DS. The MultimediaContent DS allows the description of MediaInformation, CreationInformation and UsageInformation for the source. In particular, MediaInformation which is also used to describe a variation, includes MediaProfile information, which may be useful for selecting among the variations. The MediaProfile includes MediaFormat, MediaInstance and MediaQuality descriptions which contain format details like file size, a URI indicating where the resource is stored, and quality rating information such as SNR values.

3.2. VariationSet trees

As explained in Section 3.1, the VariationSet DS allows the definition of different bindings, i.e., variations, to the same source resource. However, this concept does not directly support multiple adaptations of media content, which might be useful for some applications and scenarios. For instance, as will be further discussed in Section 5, in a proxy server (multimedia cache node), it is useful to adapt a source content several times (until the permissible quality level is reached) before replacing it from the cache. Therefore, we introduce a more general concept than the VariationSet, called the VariationSetTree.

A VariationSetTree is a tree of VariationSet DSs, where nodes represent multimedia resources. The root node is the source resource and child nodes are variation resources. The edge from one node to its child represents a Variation, and each tree level contains as many VariationSets as the number of nodes in the previous level. Leaf nodes are final resources which are no longer required to be adapted.

Fig. 3 shows an example of a VariationSetTree. The source is adapted to two variation resources: Resource 1 and Resource 2. Resource 1 is in turn adapted to two further variation resources (Resource 3 and Resource 4), whereas Resource 2 is adapted only to one (Resource 5). The VariationSetTree consists of three VariationSets (one at level one and two at level two).

The representation of a VariationSetTree within MPEG-7 is possible by sequentially encoding the VariationSets present in a tree. For instance, the VariationSetTree of Fig. 3 would lead to the MPEG-7 document shown in Fig. 4.

The intra-document references of source and variations must be realized. For instance, the source of Variations 3 and 4 is Resource 1 which is also the variation resource of Variation 1.
However, the source specification in a Variation DS does not supply a reference type, but a child of it does: the VideoType provides a MediaInformationRef which may reference to a MediaInformation already specified.

Fig. 5 shows parts of an MPEG-7 VariationSetTree for adapting the source video slovenia.cmp as described in Section 4.2. There are two levels in the VariationSetTree. In each level one variation content (derived by temporalReduction) is specified (VARIATION 1 in level 1 and VARIATION 2 in level 2). Note that the element <MediaInformationRef idref="VARIID‘/⟩ in the second-level description of the MPEG-7 document of Fig. 5 references the MediaInformation of the first-level variation specified by id="VARIID‘.

It has to be noted that in Version 2 of MPEG-7, it will be possible to specify an element SourcevariationRef of ReferenceType instead of a complete source definition to enable a reference to a source defined as a variation content earlier in the MPEG-7 description. This facilitates the definition of a VariationSetTree. However, the principal architecture of the MPEG-7 document for a VariationSetTree, as depicted in Fig. 4, remains the same.

3.3. Multimedia meta-database

MPEG-7 documents that are created as a result of applying variations to a source resource are stored in a multimedia meta-database. For the user, it serves as an entry point to the distributed multimedia architecture. It maintains all descriptive data (metadata) in the distributed architecture. Besides descriptive data pertaining to the content of the multimedia resources, the database equally handles descriptive data on the variation (quality adaptation) capabilities of these resources which are extracted by a dedicated server component, the Processing Unit, to be described in Section 4. Therefore, the adaptation information becomes a “first-class citizen” in the distributed environment from the beginning, i.e., from the query stage. Thus, a user queries the database not only for content, but also for adaptivity.

The multimedia meta-database is implemented as an MPEG-7 Multimedia Data Cartridge. A Multimedia Data Cartridge uses the new data cartridge technology of Oracle which allows plug-in system extensions into an Oracle database management system (DBMS). The data model and the indexing schema are not directly related to

Fig. 4. MPEG-7 description of VariationSetTree of Fig. 3.
Fig. 5. MPEG-7 description of Fig. 4 with intra-document references.
adaptivity and are not further described here. More information on our MPEG-7 Multimedia Data Cartridge can be found in [14,4]. The consumption of the meta-database is explained in Sections 5 and 6.

4. Processing unit: extracting VariationSetTrees

This Section provides the functionality and architecture of our implementation of metadata generation. It also presents the results of experiments that were carried out to validate our proposals.

4.1. Functionality and architecture

The Processing Unit is a supplemental server component (in addition to the meta-database and the media server) which is responsible for extracting the VariationSetTrees as MPEG-7 documents. It is situated between the media server and the meta-database and shall extract the necessary variation capability information from the audio-visual streams. This information is stored as MPEG-7 documents in the meta-database and is delivered to the components on the delivery path to the client. When a video is inserted into the database (this could be done on demand or on a regular basis), the Processing Unit shall apply operations such as transformation and scaling to the MPEG-4 encoded video, reports the results (performance and variation information) to the meta-database, and writes back the adapted video onto the video server.

The architecture of the Processing Unit is composed of a set of analysis modules. A screen shot of the Processing Unit is shown in Fig. 6.

---

**Fig. 6.** A screen shot of the Processing unit.
An MPEG-4 video is first demultiplexed into its Elementary Streams (ES). For each type of an ES (e.g., visual, audio, scene description), a series of analysis tools are defined. At the moment, analysis modules based on the variation relationships of temporal scaling, spatial scaling, and color reduction as well as all possible combinations of them are implemented. In this paper, only the results of temporal scaling by frame dropping are presented.

The analysis is carried out in two stages. First a series of analyses are done in the compressed domain, such as temporal scaling. Once the analysis of an ES in the compressed domain is completed, each ES is decompressed and a chain of analyses is applied in the decompressed domain. This is due to the observation that many variation descriptions may conveniently be extracted only in the uncompressed domain. Examples are descriptions for spatial scaling or key frame extraction. These tools rely on an Effects Plug-In chain of JMF (Java Media Framework) [25].

4.2. Results of experiments

This section gives a flavor of the results of experiments we performed. We present the results of experiments carried out on six selected MPEG-4 videos taken mostly from the MPEG-4 Reference Software [ftp://ftp.tnt.uni-hannover.de/pub/MPEG/video/conformance/version_2].

4.2.1. Original MPEG-4 encoded files

A total of six MPEG-4 encoded files were considered for this experiment. Some statistics were extracted. Table 1 shows a summary of their original size in bytes, the number and size in pixels of frames, and the recommended frame rate. For the first file, the video object elementary stream (track) was extracted from the MP4 file using the MoMuSys part of the MPEG-4 Reference Software [12].

Fig. 7 shows screen shots (first frame) of each of the video files. Note that the aspect ratio is not maintained.

4.2.2. Results of adaptation by frame dropping

I-frames and/or P- and S-frames (VOPs in MPEG-4 terminology) were extracted from these files. Video adaptation was performed in two stages: (1) dropping every other B-frame and (2) dropping the remaining B-frames. The new file sizes were recorded and the resulting file size ratios, $s_j^{(i)}$, were computed. The procedure for computing the fidelity of a variation (thus, the resulting quality $q_j^{(i)}$) is taken from [10]. It is based on the media attributes of the variation and source audio-visual data as follows. Let $a$ denote the source resource (original video) and $b$ the variation resource. Then the quality (fidelity) of $b$ with respect to $a$ is determined by

\[
\text{fidelity}(a, b) = \frac{1}{7} \left( \frac{b.\text{hasVideo}}{a.\text{hasVideo}} + \frac{b.\text{hasAudio}}{a.\text{hasAudio}} + \frac{b.\text{DataSize}}{a.\text{DataSize}} + \frac{b.\text{FrameRate}}{a.\text{FrameRate}} + \frac{b.\text{SampleRate}}{a.\text{SampleRate}} + \frac{b.\text{SpatialSize}}{a.\text{SpatialSize}} + \frac{b.\text{Colors}}{a.\text{Colors}} \right),
\]

where $\text{hasVideo}$ and $\text{hasAudio}$ are binary values ($\in \{0, 1\}$) and it is assumed that the source video has both a video and an audio track, i.e., $a.\text{hasVideo} = a.\text{hasAudio} = 1$.

The number of frames dropped was recorded and the percentages of the dropped frames were
calculated. Finally, the new frame rate that would allow the video to play without change of playback duration was calculated. The results of stage (1), dropping of every other B-frame, are shown in Table 2; and those of stage (2), dropping of the remaining B-frames, are shown in Table 3.
As can be seen from Tables 2 and 3, for the first variation level, the average file size ratio (reduced files size compared to original file size) is 0.83, which is an average file size reduction of 17%; for the second variation level, the average file size ratio is 0.65 which translates into an average file size reduction of 35%. This is a significant reduction for alternating or full B-frame dropping. Correspondingly, the resulting quality reductions are 24% and 46%, respectively. However, these are quantitative measures and may not measure the real reduction in visual perception quality.

As explained earlier in Section 3, Fig. 5 shows parts of an MPEG-7 document for adapting the source video slovenia.cmp. The consumption of this metadata document is explained in the next two Sections.

5. Metadata driven adaptation in a proxy cache

In this Section, we will show how the MPEG-7 Variation DS, discussed in Section 3, can be used in connection with quality-based video caching at a proxy server. Quality-based caching can be regarded as partial caching in the quality domain, as opposed to the usual partial caching in the time domain. Although initially whole videos are stored at the proxy, their qualities can be changed according to some criteria. This measure is, for example, beneficial to enrich the cache replacement strategy of the proxy.

We distinguish two cases: (1) quality reduction, i.e., the proxy only reduces quality; (2) quality adaptation, i.e., the proxy can reduce or enhance the quality. We will focus on quality reduction because it allows simple cache replacement strategies. Although full quality adaptation is seen as the more flexible approach, quality enhancement, however, introduces additional complexity. To enhance the quality of a reduced video, a cache has to reload specific parts of the video. Furthermore, the cache has to implement intelligent adaptive behavior. Short fluctuations in the popularity have to be filtered out to avoid oscillation of storage allocation.

Quality reduction can be effective in many cases. On the one hand, most of the videos will have a short period of high popularity followed by decreasing popularity; quality reduction supports this process. On the other hand, quality reduction can be coupled with reloading of videos; e.g., a user can trigger a reload if he/she is not satisfied.

5.1. Usage of metadata

A proxy cache can benefit from a VariationSet description to enhance its cache replacement strategy. When there is a need to replace a video in the cache, the proxy applies a quality reduction to the video, thus reducing its data size, instead of deleting it.

In the actual implementation, we rely on a single-branched version of a VariationSetTree with its depth limited to two, i.e., only two variation steps are considered. This is a realistic assumption for many adaptation algorithms. An example is given by the MPEG-7 document in Section 3, describing temporal scaling of an MPEG-4 video.

The proxy relies on the information in the MPEG-7 VariationSet DS in order to enhance its replacement strategy. Two pieces of information are actually required for executing a quality reduction step: the resulting size and the resulting quality. As presented in Sections 3 and 4, for a source video $i$, the resulting size is denoted by $s_{i}^{(j)}$ and the resulting quality by $q_{i}^{(j)}$ where $j$ indicates the level of the variation in the VariationSetTree. Both $s_{i}^{(j)}$ and $q_{i}^{(j)}$ take values in the interval $[0,1]$.

For example, the values $s_{i}^{(1)} = 0.5$ and $q_{i}^{(1)} = 0.8$ mean that we refer to a first-level variation which acts on a source video $i$ and we produce a variation video that has 50% of the original size and 80% of the original quality. The resulting quality is taken from the fidelity attribute associated with the variation.

The size and quality information may be obtained by simple XPath expressions over the MPEG-7 document.

$q_{i}^{(j)}$ for an MPEG-7 document with a source video $i$ and a variation level $j$ is calculated as follows (we assume that the variation is identified with id="Variation j"):
\[ q_i^{(0)} = \text{//Variation[@id = Variation]} /@fidelity \]

\[ s_i^{(j)} = \text{//Variation[@id = Variation]} /@fidelity \]

\[ s_i^{(j)} \text{ is calculated as} \]

\[ s_i^{(j)} = \text{//Variation[@id = Variation]} /FileSize/text() /Source[1] /FileSize/text() \] .

There actually exist many efficient implementations of XPath engines, for instance those from the Xalan-J2 XSLT processor [http://xml.apache.org/xalan-j/] and from the Saxon XSLT processor [http://saxon.sourceforge.net/].

5.2. Quality reduction and replacement

Quality-based caching is based on some assumptions. First, videos should allow quality adaptation. A video should have a number of quality steps that can be obtained through operations on that video. Such quality steps can be realized through layers (base layer and enhancement layers as defined, for example, in MPEG-2 or H.263) or through object-based coding (as defined, for example, in MPEG-4). In the following, we do not assume a special coding technique but rely on the fact that a video has a certain number of quality steps.

The second assumption is that the server provides us with an MPEG-7 description (as shown in previous sections) that describes the possible quality steps. This description is used in the replacement process. The replacement strategy is a major performance factor of proxy caches. There exist a lot of proposals for replacement strategies, e.g. [3,26]. These strategies are suitable for normal Web caches and do not introduce quality awareness in the replacement process. In the following, we will concentrate on a simple enhancement of the LRU policy, namely LRU-C (LRU with Combined Replacement), that uses quality reduction and MPEG-7 descriptions to improve the replacement process.

We assume that the videos (data about the videos) are stored in a list. For both algorithms (LRU and LRU-C), the list is sorted by recency. The most valuable video (in terms of recency) is at the beginning and the least valuable at the end. This is ensured through correct insertion, i.e., upon a request (miss or hit), a video is inserted at the beginning. The replacement starts at the end of a list when an insertion of a new video has caused an overflow, e.g., the size of all cached videos is above a given mark (95% of the cache size, for example). LRU tries to remove videos until the size of the cached videos is below a given mark. LRU-C tries to reduce the quality of videos and only eventually removes some videos. The quality reduction is implemented by eliminating one quality level in each replacement step. To simplify the design of the algorithm and to ensure a fast replacement, we use a so-called replacement pattern. This replacement pattern is a combination of two elementary replacement patterns, namely horizontal replacement and vertical replacement. (For a discussion of such patterns see [23].) For the following example, we assume that each video consists of three quality steps. The size and quality of the different steps are described by means of the MPEG-7 description from Section 3. For example, the original video \( i \) might have size \( s_i^{(0)} = 1 \) and quality \( q_i^{(0)} = 1 \), the next quality step \( s_i^{(1)} = 0.75 \) and \( q_i^{(1)} = 0.9 \) and the last quality step \( s_i^{(2)} = 0.5 \) and \( q_i^{(2)} = 0.7 \). During replacement, we use horizontal replacement for the highest layer (lowest priority). The other layers are replaced by vertical replacement. The pattern is given in Fig. 8.

This pattern is used in each replacement run. It is possible that different videos have different numbers of quality steps. The replacement algorithm tries to follow the given pattern. It is like a matrix traversal where the dimensions are given by the number of videos and the maximum number of quality steps. In each step, the replacement algorithm removes a quality step.
If it is not available, the replacement process continues with the next given quality step. This pattern is similar to patterns proposed in [24]. There, such patterns are used for the segments of a layered coded video. To be faster, we use a coarse-grain replacement and use such a pattern for the quality steps of all cached videos.

5.3. Evaluation

For the evaluation of the replacement strategies we used simulation. To have enough flexibility, we used synthetic request sequences generated by a synthetic Web proxy workload generator (ProWGen) described in [1]. We generated 50 request sequences and used 10 different Zipf values (0.1–1.0) for the popularity. A Zipf value close to 1 means skewed popularity, i.e., few popular videos. Each of these popularity values was combined with five different request sequence lengths (10,000, 20,000, 30,000, 40,000, 50,000). Each request sequence included 1000 different videos, 100 videos were one-timers (only one request). For the size of the requests, the generator uses two distributions: a lognormal for the body of the request size distribution and a Pareto distribution for the tail. Our parameters were $\mu = 7000$ kB, $\sigma = 11,000$ (lognormal) and characteristic parameter 1.2 (Pareto). We assumed no correlation between size and popularity.

Both algorithms used these sequences with a fixed cache size of 1 GB. This size was approximately 10% of the size of all videos. Therefore, each simulation run for LRU and LRU-C produced 50 values.

In terms of hit rates, the quality-based algorithm produces a higher hit rate over the whole range of the request sequences. This is due to the higher number of videos that can be stored in the cache when quality reduction is used.

Quality-based caching introduces a trade-off. On the one hand the hit rate increases, on the other hand the average quality per hit is lower than the average quality with a normal replacement strategy like LRU. For LRU, the average quality per hit is always 1. For LRU-C, the average quality is shown in Fig. 9. The average quality per hit is almost 0.9, which means that most of the videos are stored with two quality steps.

We propose to integrate this loss of quality into the hit rate by multiplying the hit rate with the average quality. This so-called quality-weighted hit rate for LRU-C is also shown in Fig. 9. Note that for pure LRU, the quality-weighted hit rate corresponds to the hit rate.

To illustrate the difference between LRU and LRU-C, we show the difference (LRU-C minus LRU) in hit rate and quality-weighted hit rate in Fig. 10.

![Fig. 9. Quality impact for LRU-C.](image-url)
From these two plots, we can draw the following conclusions:

- The improvement in hit rate is very similar over all request sequences. The improvement decreases only for high values of $a$ and longer request sequences.
- For the quality-weighted hit rate, we see a different picture. For smaller values of $a$ (less locality), LRU-C achieves a better performance (about 4% improvement). With increasing $a$, this improvement vanishes and for high values of $a$, LRU-C achieves a lower quality-weighted hit rate. This tendency is very similar for all request sequence lengths.

When $a$ is near 1, the locality of reference is very high, and therefore the normal LRU algorithm is sufficient. LRU-C tries to reduce the quality while increasing the number of videos held in the cache. This increase does not significantly improve the hit rate in this case, as most popular videos are already in the cache without quality reduction, but lowers the quality of the cached videos.

6. The client side: adaptive presentation interface

The upcoming use of multiple client devices (e.g., PCs, PDAs, mobile phones) makes it necessary to think about the adaptation of the presentation interfaces according to the terminal capabilities as well as user preferences. In order to serve multiple types of client devices, the media is presented through an adaptive presentation interface.

In this context, adaptation means the dynamic adjustment of the interface to the client’s terminal capabilities (e.g., different hardware characteristics including buffer for video players, software such as browsers and corresponding multimedia plug-ins) and the user’s preferences (e.g., user prefers to receive only images and no video). Terminal capabilities are described by means of the Composite Capabilities/Preferences Profile (CC/PP) which is a standardized framework developed by W3C, based on the RDF format [http://www.w3.org/Mobile/CCPP/]. The use of RDF enables a standard way of exchanging information about the client’s capabilities. A CC/PP standard implementation, DICE [17], was extended and integrated into the client/server environment to adapt the query and presentation interfaces to the respective usage environment descriptions.

Fig. 11 shows a screen shot of the adaptive presentation interface realized. On the right-hand side, the presentation includes video and audio. On the left-hand side presentation, the user does not have enough resources (e.g., buffer for video display), thus, only the audio track is played.
7. Summary and further work

In this paper, we have discussed the necessity of a global view on adaptation, incorporating all components of a distributed multimedia system, and introduced our project ADMITS that treats adaptation in a comprehensive way. It was shown that MPEG-7 provides tools, such as the variation description scheme, which enable the components to communicate adaptation metadata in a standardized form. By means of the proxy cache example, we have also shown how this kind of metadata can be utilized, in that case to enhance the replacement strategy of a video cache.

Work in progress includes the usage of adaptation descriptions in “active” routers and the use of offensive adaptation with the help of a virtual video server.

The results gained from the ADMITS experiments are intended to eventually provide hints on how to effectively steer and efficiently perform adaptations in distributed multimedia systems.

Acknowledgements

The authors would like to thank the anonymous reviewers for their invaluable comments.

References
