ELSEVIER

Signal Processing: *Image Communication* 15 (2000) 299–320

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

MPEG-4 Systems: Elementary stream management

C. Herpel[a,*], A. Eleftheriadis[b]

[a]*THOMSON multimedia, Karl-Wiechert-Allee 74, 30625 Hannover, Germany*

[b]*Dept. of Electrical Engineering, Columbia University, 500 West 120th Street, MC 4712, New York, NY 10027, USA*

Abstract

We describe the elementary stream management (ESM) facilities provided by MPEG-4 Systems. Within the extensive set of tools defined by MPEG-4, the ESM tools play a critical role in joining several building blocks together. ESM provides a dual to the scene description language (BIFS) in that it links the streaming resources of a presentation to the scene. We also describe the synchronization functionality as well as the system decoder model that defines the timing behavior and buffer resource management of MPEG-4 receivers. The paper concludes with considerations on data packaging in underlying delivery layer protocols and a description of the MPEG-4 content access procedure. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: MPEG-4; Object descriptor; Synchronization; Multiplex; Content access; Content delivery

1. Introduction

MPEG-4 is the first standard that views multimedia content as a set of audio–visual objects that are presented, manipulated and transported individually. This is achieved by a set of tools defined in several parts of the standard. First, the media compression schemes defined in the Visual [5] and Audio [6] parts of the MPEG-4 specification are object oriented so that they can represent elemental audio–visual entities (e.g., arbitrarily shaped visual objects). The overall architecture of MPEG-4 that provides the means for the combined use of these elements is defined in MPEG-4 Systems [4]. The *scene description* language, called Binary Format for Scenes (BIFS), discussed in a companion paper in this Special Issue [19], expresses how individual audio–visual objects are to be composed together for presentation on the user's screen and speakers. In addition, a *stream description* of the corresponding elementary streams that convey this data is needed. The term *elementary stream management* is used to refer to the entire set of functionalities needed to describe, express relations between, and effect synchronization among such data streams. In addition to describing these tools, we also provide some considerations about data packaging and multiplexing on the underlying delivery infrastructure.

The organization of this paper is as follows. First, some terms that are used throughout the MPEG-4 specification as well as in this paper are briefly introduced and discussed. Then, we describe the object description framework that allows the identification and characterization of a related set of elementary streams. It serves as a guide through the

* Corresponding author.

E-mail address: herpelc@thmulti.com (C. Herpel)

0923-5965/00/$ - see front matter © 2000 Elsevier Science B.V. All rights reserved.
PII: S0923-5965(99)00051-X

potentially large set of streams that may belong to a sophisticated MPEG-4 presentation. This guide must be first consulted before any of the content conveyed within the streams can be presented. Next, the issue of synchronization is discussed. Since many streams are quasi-concurrently generated and consumed, time stamping is employed here, similar to MPEG-2, to align them on the temporal axis. In this context, the system decoder model is introduced; it describes in an abstract way how the notion of time is defined in MPEG-4 and how some of the buffer memory in the system is modeled and managed. Finally, some ongoing activities will be reported concerning delivery of MPEG-4 data. The standard does not specify its own transport layer, but is intended to be used with different delivery systems (including digital TV broadcasting, the Internet, and stored files). Delivery functionality in general is described in depth in a companion paper in this Special Issue [8]. Equipped with these building blocks, we finally present a walkthrough of the process of accessing MPEG-4 content consisting of many different elementary streams, possibly even originating from different locations.

2. Fundamental concepts

Audio or visual entities that participate as individual elements in a scene are termed *audio–visual objects*. Such objects can be either natural or synthetic. Time-variant data for each natural object is conveyed separately, in different "channels". Synthetic objects may be generated with the graphics and synthesized sound operations provided by BIFS. BIFS is actually more than just a scene description language, in that it integrates both natural and synthetic objects in the same composition space. Some objects may therefore be fully described within the scene description stream itself. Objects may also be animated using the BIFS-Anim tool. In that case, the corresponding coded information will as well be conveyed in its own channel. As a result, the term audio–visual object – although conceptually clear – cannot be uniquely associated with just one feature or syntactic element of MPEG-4.

As a general paradigm in MPEG-4, all information is conveyed in a streaming manner. The term *elementary stream* refers to data that fully or partially contains the encoded representation of a single audio or visual object, scene description information, or control information. In other words, elementary streams are the conceptual delivery pipes of MPEG-4; they are mapped to actual delivery channels using mechanisms that are described in detail later on.

Elementary streams, or groups thereof, are identified and characterized by *object descriptors*. This includes the scene description, audio–visual objects data, as well as object descriptor streams themselves. The information contained will include the format of the data (e.g., visual face animation stream) as well as indication of the resources necessary for decoding (e.g., profile/level indication). Alternate representations or scalable encodings using multiple streams for a single audio–visual object can also be signaled by the object descriptor. An object descriptor may, for example, specify a set of elementary streams that jointly contain the compressed representation of an audio signal. Most importantly, it will contain the information necessary to directly or indirectly identify the location of the data, either in terms of an actual transport channel or a URL. An indirect approach is necessary so as not to use network and/or protocol-specific addressing.

Scene description and stream description are strictly separated in MPEG-4. In particular, the scene description contains no information about the streams that is needed to reconstruct a particular audio–visual object, whereas the stream description contains no information that relates to how an object is to be used within a scene. However, it is a convenient place to add helpful meta-information to streams. As we will see, even the stream description itself is conveyed in the form of elementary streams. The separation facilitates editing and general manipulation of MPEG-4 content. The scene description is usually consciously authored by the content creator while the stream description mostly follows from general content creator preferences, default settings of an editing tool, or even service provider constraints and rules. The link between the two descriptions is a numeric object descriptor

identifer that the scene description uses to point to object descriptors, which in turn provides the necessary information for assembling the elementary stream data required to decode and reconstruct the object at hand.

The actual packaging of the elementary streams as defined by MPEG does not depend on a specific delivery technology. MPEG-4 defines a *Sync Layer* that just packetizes elementary streams in terms of *access units* (e.g., a frame of video or audio data) and adds a header with timing and other information. This is done in a uniform manner for all different stream types in order to ease identification and processing of these fundamental entities in each stream. All further mappings of the streams to delivery protocols and their control are handled by the delivery layer [8] and are to be defined outside the MPEG-4 Systems standard. For example, transport of MPEG-4 content over RTP is to be defined under the auspices of IETF.

3. Object descriptors: the link between scene and streams

3.1. Overview

A hierarchically structured set of descriptors constitutes the major building block of the object description framework. The highest-level descriptor is the object descriptor (OD) itself. It serves as a shell that aggregates a number of other descriptors. Most prominent among these are the elementary stream descriptors (ES descriptors) which describe individual streams. Additional auxiliary information can be attached to an OD, either to describe the content conveyed by these streams in a textual form (object content information, OCI) or to do intellectual property management and protection (IPMP). Finally, rules are needed on how these descriptors may be used and, more important, how the descriptors are actually communicated via the so-called OD commands. We first introduce the semantics of the descriptors themselves and later on discuss the implications that arise from the fact that they are conveyed in a streaming fashion.

3.2. The object descriptor: describing relations between streams

An object descriptor is merely a shell that groups a number of descriptive components that are linked to an MPEG-4 scene through a single node of the scene description. The major components are depicted in Fig. 1. The linking to the scene is achieved in a two-stage process. First, there is a numeric identifier, called the object descriptor ID (OD_ID), that labels the object descriptor. This identifier is referenced by the scene description stream and thus links the object descriptor to the scene. The second stage involves the actual binding of elementary streams identified and described by the elementary stream descriptors included in this object descriptor, using another identifier, called ES_ID, which is part of the elementary stream descriptor.

In the most simple case, an OD contains just one *ES descriptor* that identifies, for example, the audio stream that belongs to the AudioSource node by which this OD is referenced, as illustrated in Fig. 2(a). The same object descriptor may as well be referenced from two distinct scene description nodes (Fig. 2(b)). On the other hand, within a single OD it is also possible to have two or more ES descriptors, for example, one identifying a low bit-rate audio stream and another one identifying a higher bit-rate stream with the same content (Fig. 3(a)). In that case the terminal (or rather the

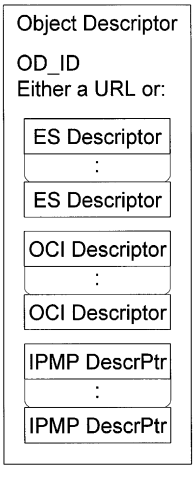


Fig. 1. Main components of the object descriptor.

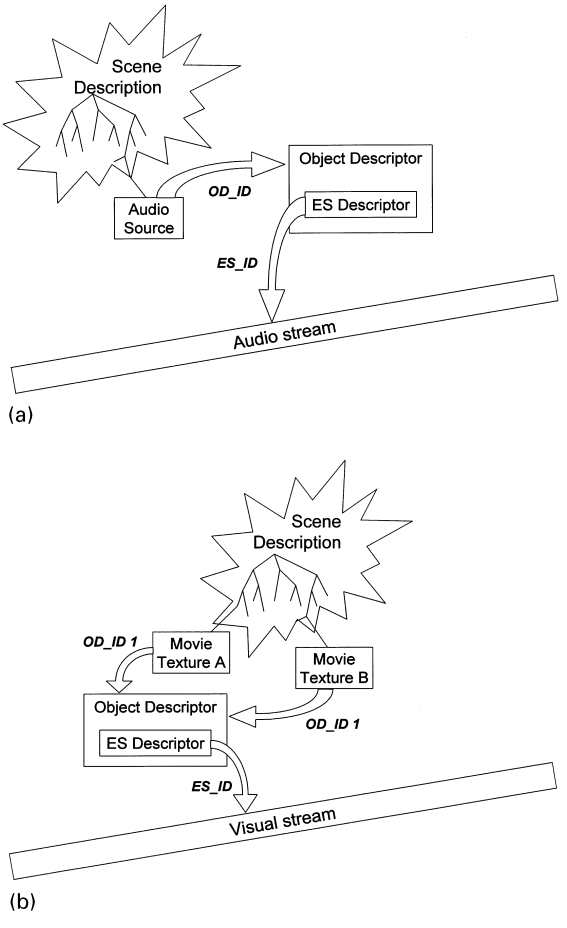


Fig. 2. ES reference through object descriptor (a) from one scene description node, (b) from two scene description nodes.

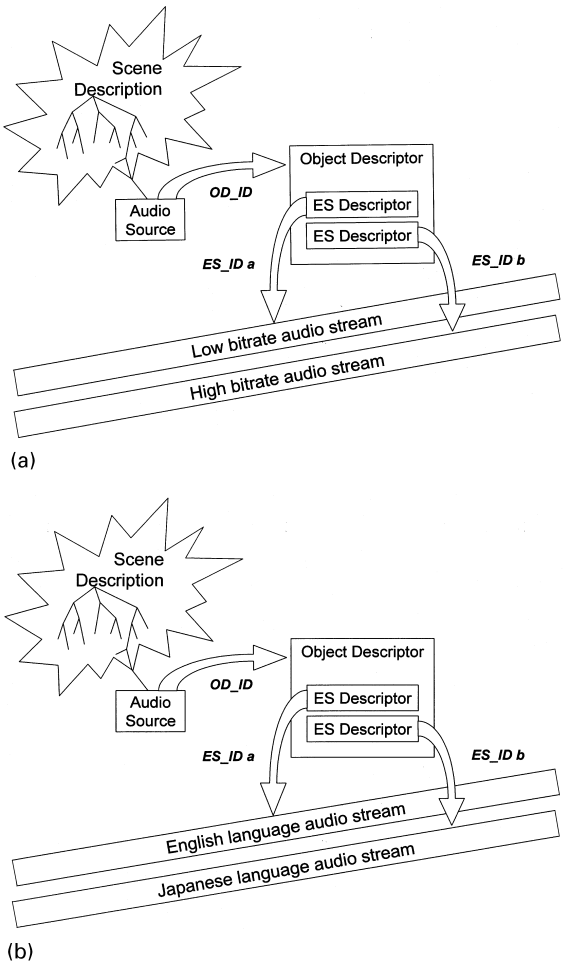


Fig. 3. Reference to multiple ESs through one object descriptor (a); audio application: multiple languages (b).

user) has a choice between two audio qualities. Specifically for audio, it is also possible to have multiple audio streams with different languages that can be selected according to user preferences (Fig. 3(b)).

In general, all kinds of different resolution or different bit-rate streams representing the same audio or visual content may be advertised in a single-object descriptor in order to offer a choice of quality. In contrast, streams that represent different audio or visual content must be referenced through distinct object descriptors. As an example, an AudioSource and a MovieTexture node that (obviously) refer to different elementary streams have to utilize two distinct ODs as shown in Fig. 4.

Finally, it is possible to describe within one-object descriptor a set of streams that corresponds to a scalable, or hierarchical, encoding of the data that represent an audio–visual object. In that case it is necessary to signal not only the properties of the individual elementary streams but also their inter-dependencies. These may be trivial as indicated in Fig. 5(a) where each stream just depends on the previous one, or more complex as shown in Fig. 5(b) where the same base layer stream is referenced by two other streams, providing, for example, quality improvements in the temporal domain (temporal scalability) and in the spatial domain (spatial scalability). The precise instructions on how the decoder has to handle each individual

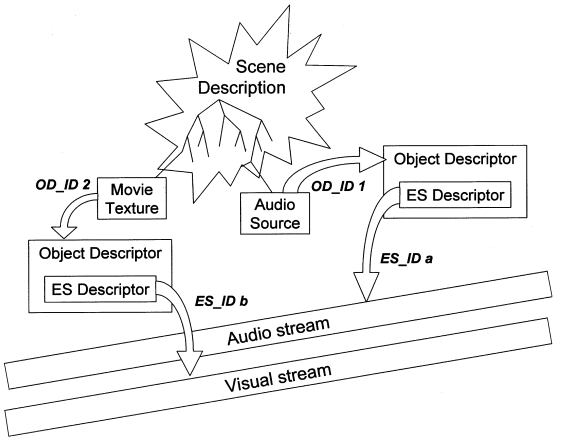


Fig. 4. Different scene description node types need different object descriptors.

stream in such a case is incorporated in a DecoderSpecificInfo sub-descriptor that is contained in each ES descriptor.

Hierarchical dependencies are only allowed to exist between the set of ES descriptors that are included in a single-object descriptor. If the same elementary stream is to be referenced from two different scene description nodes within different contexts, say, (a) as a single-quality stream and (b) as the base layer of an audio–visual object encoded in a scalable fashion, then there have to be two different object descriptors, however both of them will include the same ES descriptor pointing to the base layer stream (Fig. 6).

Additional descriptors in the OD may carry further information that refers to the whole object. Such information includes object content information and intellectual property management and protection information as detailed further below.

Instead of providing information about the media object in-place, the object descriptor may contain only a URL that points to another object descriptor at a remote location. In that case an object descriptor has to be retrieved from that location according to the rules implied by the actual URL syntax used. The content of that object descriptor will be used instead of any locally provided information. Only the object descriptor ID remains the same as before, since it is the element that is referenced by the scene description which

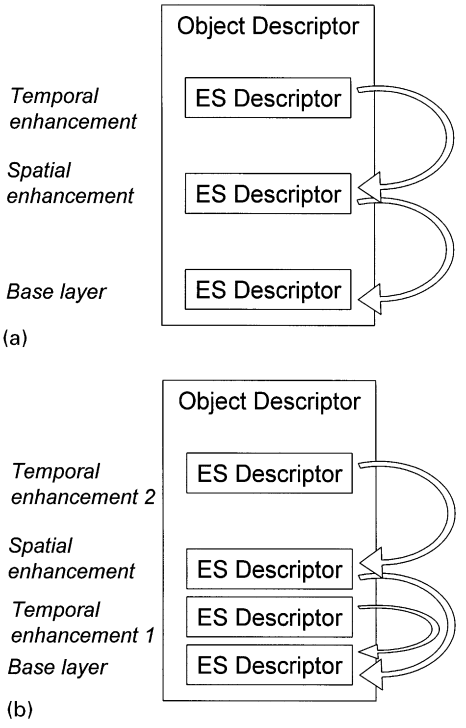


Fig. 5. (a) Simple and (b) more complex ES depending indication.

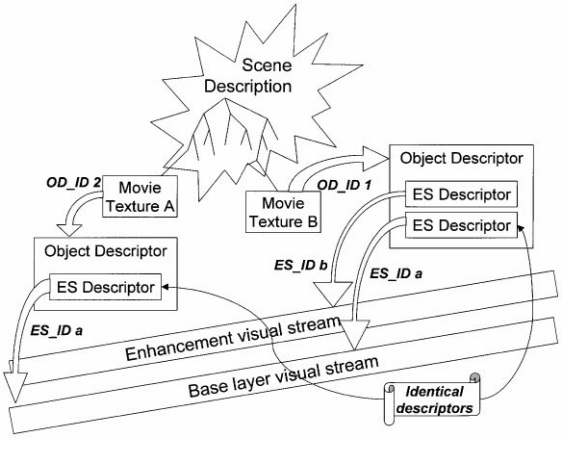


Fig. 6. Referencing the same ES in different contexts.

need not be aware of the fact that this object descriptor is retrieved from a remote location.

In order to "bootstrap" an MPEG-4 presentation, it is necessary to identify the elementary

streams that contain the scene description and the associated object descriptors. A special OD called the Initial OD (IOD) is defined in order to convey that information. In addition to the elements of the regular object descriptor, the IOD also contains information about the complexity of the overall presentation, expressed in profile and level indications. The IOD is usually communicated out-of-band, e.g., during the session initialization phase before any elementary stream channels are set up. However, since content may be hierarchically nested using the Inline node of the scene description, IODs may also occur in place of regular object descriptors in that special case, allowing to insert complexity information not only for the overall presentation but also for sub-scenes.

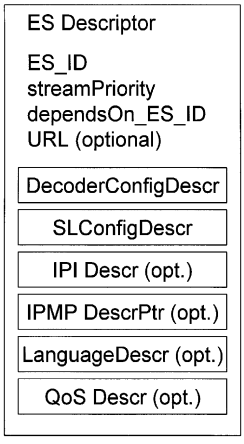


Fig. 7. Main components of the ES descriptor.

3.3. The elementary stream descriptor: all properties of a single stream

Each elementary stream descriptor contains all descriptive information available for a single elementary stream (ES), as illustrated in Fig. 7. It identifies this stream with a numeric ES_ID and an optional URL. ES_IDs allow the location of this stream by number. ES_IDs are unique within a limited, well-defined name scope (see Section 3.6). URLs are more flexible and allow referring to streams by a globally unique name. In contrast to the object descriptor, if a URL is present in an ES descriptor, this does not mean that the descriptive information about this stream is retrieved from a remote location, but rather that the stream itself can be located using this URL. All the descriptive information is still present in the local ES descriptor. Again, the reason for the presence of a URL is to facilitate the description of distributed content.

The above-mentioned indication of stream dependencies is signaled in this descriptor, as well as the stream priority which is a relative indication that may be used to group streams according to their priority at the delivery layer.

The *DecoderConfig descriptor* is a mandatory sub-descriptor of each ES descriptor and contains all information that is needed to initialize a media decoder for this elementary stream. Since an elementary stream descriptor may also describe control streams with object-related information flowing upstream, it may in fact be that the configuration of the associated control command *encoder* is described here. Note, however, that actual back channel signaling protocols are not included in Version 1 of MPEG-4.

The DecoderConfig descriptor includes an indication of an object type [14], a stream type as well as information about average and maximum bitrate and the size required for the decoding buffer in the receiving terminal.

The stream-type indication in the DecoderConfig descriptor just identifies the most basic aspects of the stream, i.e., if it is a visual stream, audio stream, scene description stream, object descriptor stream, clock reference stream or object content information stream. More detailed information about the stream is provided by the object-type indication, which defines the syntactic sub-sets of the compression scheme for this object. In the language of the visual part of MPEG-4 it should rather be called “object layer type” than “object type”. MPEG-4 defines a number of object types for both visual and audio objects.

Another important element of the DecoderConfig descriptor is the embedded decoder specific configuration information that is conceptually passed on to the media decoder selected and initialized on the basis of stream-type and object-type indication. This information corresponds to what classically is referred to as “high-level headers” in previous

audio–visual compression standards (e.g., "sequence headers" in MPEG-2 Video). This also explains why the content of this descriptor for visual and audio media types is actually specified in Parts 2 and 3 of MPEG-4, respectively.

The mandatory *SLConfig descriptor* carries configuration information for the Sync Layer (SL) of this elementary stream. Since the features of the SL are discussed in Section 4 of this paper, we defer a detailed discussion of the elements of the SLConfig descriptor to that section as well.

3.4. Auxiliary descriptors and streams

Apart from grouping media streams, an object descriptor allows to point to auxiliary streams associated to this set of media streams. There are three types of such streams. First, semantic information about the content of the audio–visual object that is fed by these streams may be conveyed by means of *object content information* (OCI). Then, *intellectual property management and protection* information may be attached. Finally, *clock reference streams* may be used to convey time base information (see Section 4). Another type of auxiliary information are the *QoS descriptors*, which may be attached to ES descriptors. And, of course, the design of the descriptors as self-describing entities constitutes a generic *extension mechanism* that may be used by ISO or specific applications to attach arbitrary descriptive information to an audio–visual object.

3.4.1. Object content information

Object content information basically consists of a set of *OCI descriptors*, that communicate a number of features of the audio–visual object that is constructed by the elementary streams associated with a given object descriptor. There is a descriptor with keywords, possibly to be used with search engines, a textual description of the content, language and content rating information, as well as creation dates and names for both the content item and the authors of this object content information.

These descriptors may be included directly in object descriptors to indicate static OCI properties. Since all elementary streams that are collected within one object descriptor are supposed to refer to the same content item, i.e., the same audio–visual object, in general there is no OCI for specific elementary streams. One exception are language descriptors which can be associated to individual elementary streams (see Fig. 7). An object descriptor for an audio object may contain a set of different language streams that are activated according to user preferences.

In case the OCI changes over the lifetime of the media streams associated to this object descriptor, it is possible to attach an *OCI stream* to the object descriptor. The OCI stream conveys a set of *OCI events* that are qualified by their start time and duration. This means that an MPEG-4 presentation may be further decomposed semantically based on the actual content presented.

OCI streams as well as OCI descriptors may be attached to either audio, visual or scene description streams. This influences their scope. When attached to object descriptors with audio or video streams OCI obviously just describes those streams, as depicted in Fig. 8(a). When attached to a scene description stream, OCI describes everything that is, in turn, described by the scene description (Fig. 8(b)). In this case indeed OCI *streams* (rather than single descriptors) may be most useful, since a scene description typically conveys a sequence of semantically meaningful audio–visual events.

Finally, it should be noted that OCI is a first step towards MPEG-7, in the sense that the expectation for MPEG-7 is that it will provide much more extensive semantic information about media content in a standardized format. It is thus likely that OCI will only be a temporary solution to be superseded or extended by the results of this new standardization effort.

3.4.2. Intellectual property management and protection

The IPMP framework consists of a fully standardized *intellectual property identification (IPI) descriptor* as well as *IPMP descriptors* and *IPMP streams* which are a standardized shell with non-normative (i.e., not standardized) content.

The IPI descriptor occurs as an optional part of an ES descriptor and is a vehicle to convey standardized identifiers for content, if so desired by the content author or distributor. Examples include

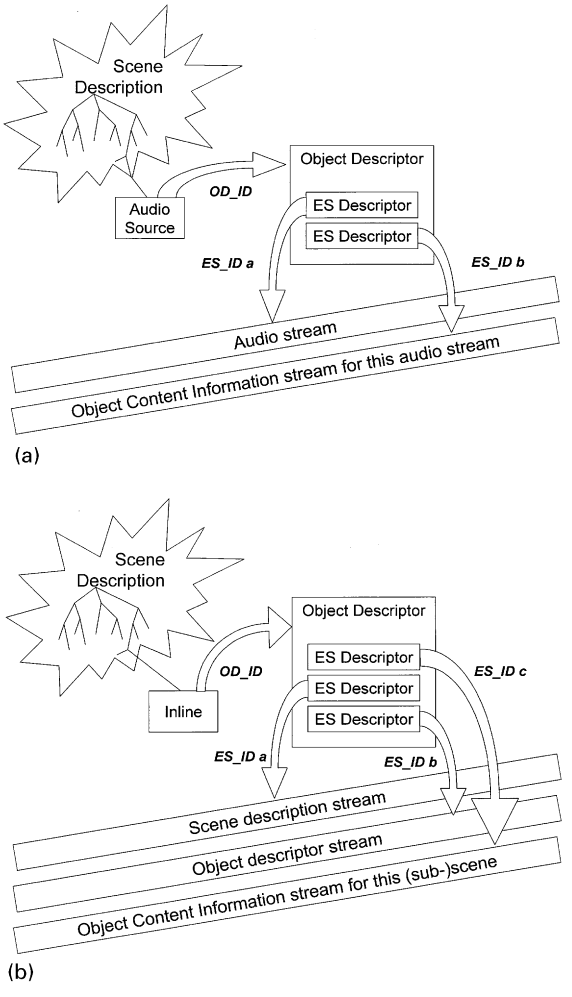


Fig. 8. Attaching an OCI stream to (a) a media stream or (b) a whole subscene.

ISBN (International Standard Book Number), ISMN (International Standard Music Number) or DOI (Digital Object Identifier). If multiple audio–visual objects within one MPEG-4 session are identified by the same IPI information, the IPI descriptor may just consist of a pointer to another elementary stream (i.e., its ES_ID) that carries the IPI information.

The core IPMP framework consists of IPMP descriptors and IPMP streams. It has been engineered in a way that should allow the co-existence of multiple IPMP systems that govern the conditional access to specific content streams or entire presentations. This has been done in acknowledgment of the fact that it is unlikely that only a single IPMP system will be used for MPEG-4 content.

IPMP descriptors convey proprietary information that may help to decrypt (content) elementary streams or that contain authorization or entitlement information evaluated by an – equally proprietary – IPMP subsystem in the receiving terminal. IPMP descriptors have IDs that may be assigned to individual vendors of IPMP systems by a registration authority, so that descriptors for different systems may be attached to content without conflict.

IPMP descriptors are conveyed in OD streams, but are not part of object descriptors or ES descriptors. IPMP descriptor pointers inside either object descriptors or ES descriptors (see Fig. 7) are used to refer an IPMP descriptor with a specific ID. If placed in an object descriptor, it means that the IPMP descriptor is relevant for all streams described by this OD. If placed in an ES descriptor, it will only be valid for this stream.

IPMP descriptors may occasionally be changed over time. However, IPMP information that needs frequent updates in a streaming fashion should be conveyed in IPMP streams. IPMP streams furthermore allow to keep IPMP information separate from the original MPEG-4 information. An example of a terminal with an IPMP system and all the possible control points is depicted in Fig. 9. The meaning of "control point" is IPMP system specific and may translate, for example, to decrypting or enabling of data flows or to reports about this data flow.

3.4.3. *Quality of service descriptor*

The *quality of service (QoS) descriptor* is an additional descriptor that may occur in ES descriptors. It aims to qualify the requirements that a specific elementary stream has on the QoS of the transport channel for this stream. The most obvious QoS parameter is the stream priority and a possibility to signal predefined QoS scenarios, most notably "guaranteed" and "best effort" channels. Apart from that, unfortunately it remains difficult to agree on a universal set of QoS parameters that is valid for a variety of transport networks. Therefore, a generic set of QoS_Qualifiers has been adopted

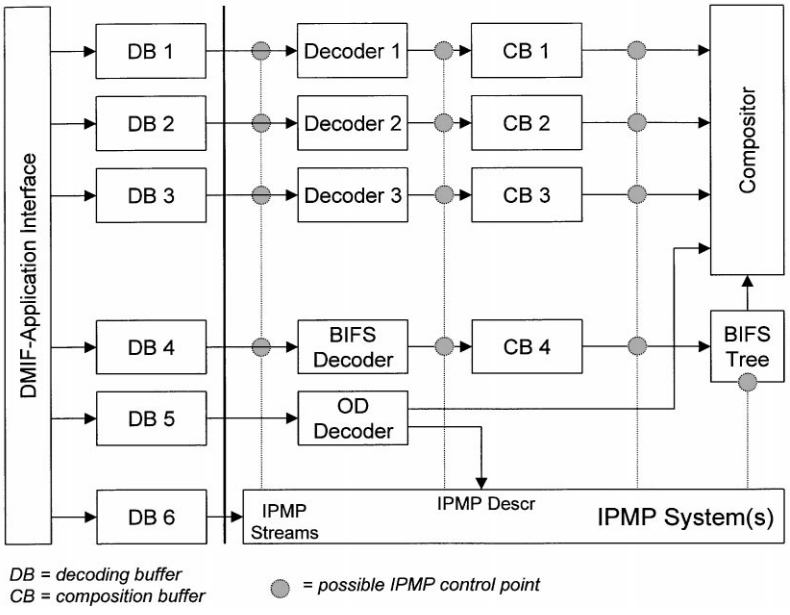


Fig. 9. IPMP system in an MPEG-4 terminal.

that needs to be assessed and possibly extended in the scope of specific applications.

QoS descriptors at the ES level have an obvious use in interactive scenarios, where the receiving terminal may select individual elementary streams based on their traffic (signaled in DecoderConfig descriptor) and their QoS requirements, as well as the associated communication cost. However, QoS descriptors are allowed as well in multicast and local storage scenarios. They may also be of interest in heterogeneous networks, if a network bridge needs to be made aware of the requirements of individual elementary streams that will, in turn, allow more intelligent processing by this bridge.

3.5. Conveying object descriptors as a stream

We now explore the process with which object descriptors actually reach their destination. It has already been mentioned that they are streamed, similarly to an audio, visual, or scene description stream. In order to provide flexibility, ODs are not just put in an elementary stream one after the other; instead, a lightweight *object descriptor protocol* has been defined to encapsulate object descriptors in *OD commands*. These commands allow for update or removal of a set of object descriptors or individual elementary stream descriptors at a specific point in time. In the same way, IPMP descriptors can also be updated or removed.

The timing aspect is very important, since time stamps on object descriptors can be used to indicate at which point in time the terminal is expected to be ready to receive data for a specific, newly set up elementary stream. The time stamp associated with such a command is placed on the Sync Layer (see Section 4), as with any other elementary stream.

Updates of object descriptors usually mean that additional elementary streams show up in the updated OD. How to use this feature is largely left to users. For example, a service provider could use this feature to communicate to the receiver that a higher bit-rate version of the same content is now available, maybe because server load or network load has been reduced enough to make this possible. There is no differentiation, however, between an OD update and sending a completely new OD. So, whenever in the middle of a presentation new audio–visual objects enter a scene, the associated elementary streams may be made known to the receiving terminal by a corresponding OD update command with the new object descriptors.

Since object descriptors as well as the scene description constitute absolutely vital information for an MPEG-4 player, it is recommended that reliable transport channels be used for both, at least in unicast applications. In multicast applications this may not always be possible. This data, however, can be periodically repeated or conveyed out of band to enable random access to the multicast MPEG-4 session. In that respect, object descriptor information is comparable to Program Specific Information (PSI) or Service Information (DVB/SI) in MPEG-2-based applications.

3.6. Relations and scoping of BIFS and OD Streams

As was discussed earlier, object descriptors and the scene description are conceptually different and therefore separated; nevertheless, a strong link exists between the two. This is also evident from the rule that an object descriptor that describes one or more scene description streams must also describe the related object descriptor streams. So, an object descriptor is the glue between both. In fact, it will be an *initial* object descriptor at the initial access point to content. Object descriptors attached to Inline nodes that also contain pointers to both object descriptor streams and scene description streams can either be initial object descriptors or ordinary object descriptors (see Fig. 10).

As long as MPEG-4 content consists of just a single-object descriptor stream, an associated scene description stream and a number of audio–visual streams as needed by the scene, it is quite easy to know the scope of OD_IDs, ES_IDs and other identifiers used by both the scene description and the object descriptor stream. There is only one single name space or scope in that case.

Now, what if there are multiple object descriptor streams? This may occur in two ways. First, the use of Inline nodes in the scene description forces additional scene description streams into existence. Also, there is usually no scene description stream without an object descriptor stream, unless the additional scene does not refer to any media streams. Second, as was discussed earlier, there may be multiple scene description and object descriptor streams that are associated within one object descriptor. In that case, there is still only one Inline node but multiple scene description streams are linked to it. The use of this scenario will be explored further later on.

For these two scenarios there is a simple name scoping rule that is inspired by the semantics of the Inline node: all scene description and object descriptor streams that are associated to a single-object descriptor constitute a single name scope for the identifiers used by them, since they will all be attached to the scene through a single Inline node. Object descriptor streams and scene description streams that are announced through different object descriptors have different name scopes, as illustrated in Fig. 11. Note again that instead of talking about an Inline node, we could as well refer to the top (or only) scene description. In that case the object descriptor in question would be the initial object descriptor.

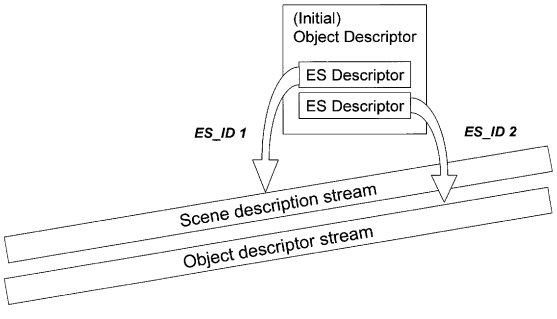


Fig. 10. (Initial) object descriptor as a reference to scene description and OD streams.

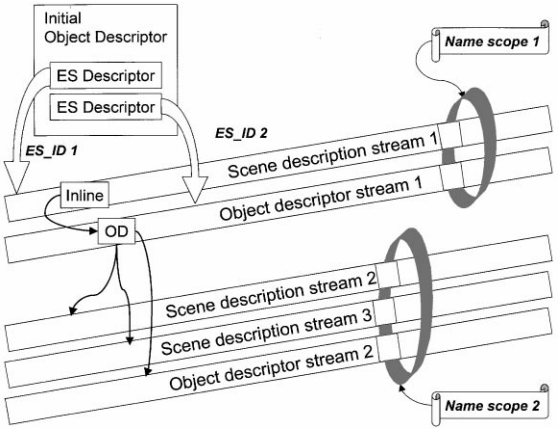


Fig. 11. Scene description and OD streams for a hierarchical scene (using Inline node).

3.7. Usage of URLs

URLs are actually a mechanism to escape the above-mentioned scoping rules. If parts of the content, e.g., complete subscenes or individual elementary streams, have well-known absolute addresses, these may be indicated with a URL in any of the places indicated above. A URL in an object descriptors identifies a service entity that will subsequently make available the content it is responsible for, following the procedures described in Section 6. In contrast, a URL in an ES descriptor just points to the location of an elementary stream. This stream is subsequently delivered to the receiver by the service entity that is associated to the current name scope.

The actual syntax of URLs will likely depend on the application scenario and delivery layer, since it may contain a fully qualified service or content address. URL syntax and semantics has to be designed carefully, for example, avoiding to put volatile transport channel indicators instead of persistent content location information in it. This is necessary, since application contexts like digital TV broadcasting might require re-assignment of transport channels on the way between sender and receiver. Currently, a number of standardization activities both in the World Wide Web Consortium (W3C) and industry groups such as the Digital Audio–Visual Council (DAVIC) try to define a broader basis for the so-called "TV URLs" for broadcast applications. This work may have relevance to the ongoing efforts to standardize URL schemes (i.e., the URL syntax) within MPEG itself.

3.8. Structuring content by grouping of streams

An interesting question is how to use grouping of elementary streams through one object descriptor to improve the structure of content. In particular, we want to be able to arrange a large number of elementary streams into groups. Even though initial MPEG-4 applications may only require a small number of streams, the design allows to efficiently manage content consisting of a large number of streams. As hardware and software capabilities improve, it is only natural that the sophistication of content will also increase.

A number of application scenarios are conceivable; among them: the delivery of differentiated quality content to different user groups, delivery of different portions of the content to different user groups and reception of content originating from different sources. Grouping is quite relevant for multicast applications where it should be easy to remove parts of the content somewhere "on the way", while in point-to-point applications it should also be possible to negotiate the desired subset of elementary streams between client and server.

Let us briefly examine how these scenarios can be handled with the object description framework.

3.8.1. Grouping streams for content partitioning or quality scaling

Content partitioning, for example providing a preview and a full version of some content, is possible without having to send any part of the content more than once. The preview could either omit some parts of the content or present it in a lower quality. In any case streams need to be grouped in a base set of streams for the preview and additional streams for the full version of the content.

First, the scene description has to be split appropriately. Parts of the scene that should go both in the preview and full versions, e.g., consisting of some audio and still images, have to form one stream. The scene description for the enhancements, e.g., additional video and a high detail mesh graphic, forms the second stream. Actually, the graphics objects can even be scalably encoded to some extent, providing graphics with different level of detail in the two scene description streams. Both scene description streams live in the same name space. Therefore, full flexibility exists to cross-reference between both parts of the scene description. The ability to cross-reference is important if the scene has interactive elements, i.e., may be manipulated by the user.

Now, two object descriptor streams can be set up, one conveying only ODs for the media streams required by the preview content (audio and still images), the other one conveying ODs for the additional media streams referenced by the complete content (video and mesh graphics). This scenario is illustrated in Fig. 12 where media streams 1 and

2 correspond to the basic content while media streams 3, 4 and 5 convey the additional information.

The example in Fig. 13 illustrates content quality scaling. Media streams 1 and 2 carry the basic, low-quality information of the content. Object descriptors in OD stream 1 just have the ES descriptors that point to those streams. The object descriptors with the same OD_IDs in OD stream 2, on the other hand, have only the ES descriptors that point to the enhancement streams conveyed in media streams 3, 4 and 5. A terminal interested in the basic quality only receives and processes the first stream group, while for the higher quality the enhancement streams have to be processed as well.

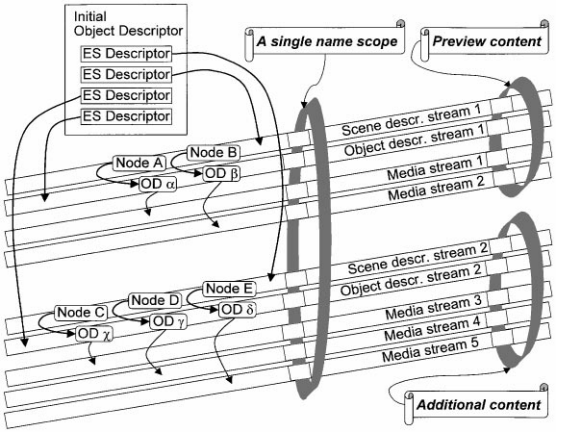


Fig. 12. Object scalability with a single name scope.

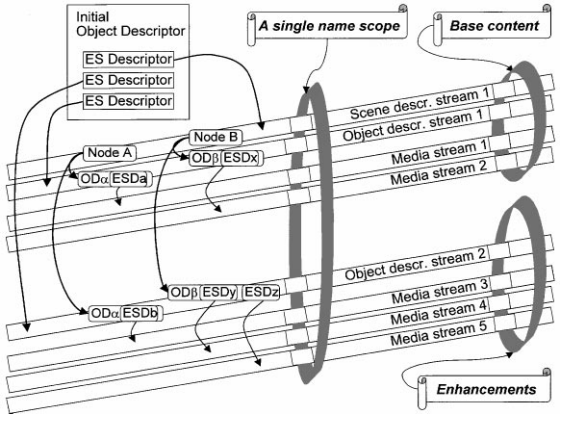


Fig. 13. Content from different locations.

3.8.2. Grouping streams for location

A special kind of content partitioning occurs in applications that present content originating from various locations at one receiving terminal. The most obvious application here is a video conference. In that case, usually Inline nodes would be used in the scene description since the individual sub-scenes are essentially independent and do not need to share name spaces for their OD_ID and ES_ID identifiers. Obviously, there will be at least one object descriptor stream and one scene description stream plus the needed media streams originating from each content source, as shown in Fig. 14 (with some more abstraction for better readability). Apart from that, this scenario is quite similar to the previous one where the different portions of content come from the same source.

3.8.3. Handling of grouping information on the delivery layer

The grouping relations expressed through object descriptors have to be evaluated by a server in order to appropriately multiplex the streams together. This may be facilitated by the definition of appropriate QoS_Qualifiers in QoS descriptors. In case of quality scalability, for example, the basic quality streams could be transmitted in a more reliable channel than the enhancement, in order to improve the resilience to transmission errors.

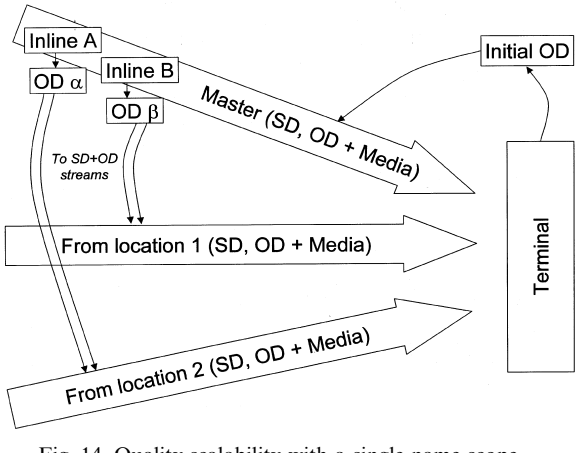


Fig. 14. Quality scalability with a single name scope.

It may be desirable that MPEG-4 agnostic service agents in a network are enabled to easily remove portions of the content, e.g., if there is not enough bandwidth available or if the user does not wish to pay for them. Therefore, the multiplexed bitstreams need to be flagged appropriately. Such procedures depend on the actual delivery layer and are therefore not in the scope of the MPEG-4 Systems standard.

3.9. Managing content complexity

It was already mentioned that initial object descriptors convey some indication of profiles and levels of the content referenced by them. These scalar indications allow an immediate decision by the receiving terminal about whether or not it is able to decode and present the content being received. Due to the potential complexity in terms of the number of scenes, it has also been made possible to indicate such complexity only for "the current subscene", i.e., excluding parts of the scene that are included through Inline nodes.

In the absence of profile and level indications or at the discretion of the receiving terminal, it is also possible to evaluate the decoder configuration, stream priorities, bit-rate requirements and the dependencies signaled in the ES descriptors. This allows the receiving terminal in an interactive application to request the delivery of a meaningful subset of streams for each media object, so that the computational resources of the terminal are not exceeded. Apart from these resource-driven considerations, the terminal or the user needs, of course, to evaluate the scene description in order to decide which subset of the media objects advertised in an object descriptor stream are relevant.

3.10. Object descriptor usage for multicast of MPEG-4 content

Multicast or broadcast applications are characterized by the fact that clients may randomly tune into a running presentation. To accommodate this, usually all the crucial information for the configuration of elementary stream decoders is periodically repeated. Such a classical (e.g., MPEG-2 like) scenario is depicted in Fig. 15(a) and shows that each stream follows its own schedule for inserting some configuration information.

In MPEG-4 the object descriptors convey the basic information that is required to gain access and be able to process the media streams that form part of an MPEG-4 presentation. The

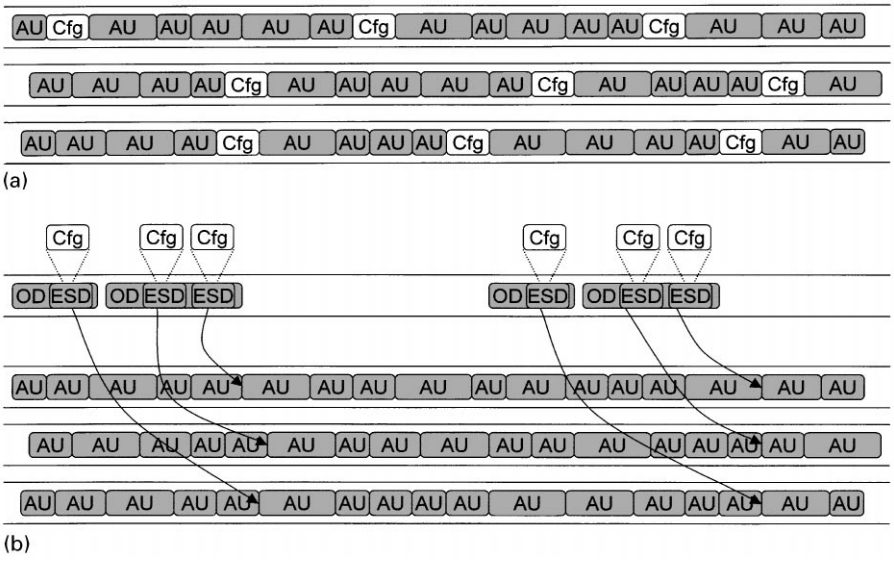


Fig. 15. Periodic repetition of configuration info for broadcast; (a) MPEG-2 approach and (b) using object descriptors.

configuration information is part of the DecoderConfig descriptor and, therefore, is automatically repeated if the ODs themselves are repeated. This repetition is necessary in both broadcast and multicast applications as shown in Fig. 15(b).

Even in a multicast scenario it is recommended to convey the object descriptors on a rather reliable channel, which has the positive side effect that crucial decoder configuration data might be received more reliably than the media data.

3.11. Distributed content handling considerations

Similar to MPEG-2, MPEG-4's architecture is based on an abstract, idealized receiver model. This allows precise resolution of issues that have to do with performance, timing and delay, and so on. In practice, implementers make sure that their designs approximate this idealized design or, more appropriately, that their implementations compensate for the imperfections of the environment on which they operate.

In contrast with MPEG-2, however, MPEG-4 allows the construction of content where different parts may originate from different locations. Furthermore, MPEG-4 makes no assumptions about the type of the underlying communications infrastructure (IP, ATM, broadcast, etc.). It is then impossible to fully ensure that use of MPEG-4's distributed content capabilities will result in seamless content presentation in all circumstances.

We expect that content creators, jointly with both content and communication services providers will create content cognizant of the environment on which it will operate, leveraging the available resources in order to provide a maximum-quality user experience. This is similar to the practice of tuning of HTML pages, so that they provide both visually rich content as well as reasonable download times over telephone lines. At any rate, the expectation is that the evolution of the Internet and other suitable transport networks will gradually solve these issues.

3.12. Authoring stream and scene descriptions

Scene descriptions are intended for manipulation only by content creators and associated software tools. As a result, the encoding of scene descriptions is quite sophisticated, and is based on extensive node coding tables. In contrast, stream descriptions are not only created by content creation tools but they can also be modified by service provider software such as servers or gateways. A simple example where this is necessary is when a server performs real-time multiplexing or re-multiplexing of a set of streams, in which case the ES_IDs may have to be rewritten. Another example is "cookie" generation. To facilitate this type of processing, all descriptors follow simple byte-oriented and byte-aligned encoding, so that the computational overhead is minimized. We should point out that the multiple levels of indirection that are found in the various parts of MPEG-4 Systems are instrumental in facilitating these types of operations. For example, it is possible to remultiplex streams *without* modifying any bit of the content, by only reorganizing the mapping of ES_IDs to actual transport layer channels (stream map table).

4. Elementary stream synchronization

The synchronization of elementary streams is accomplished by the well-known concepts of time stamps and clock references, as used for example in MPEG-2 Systems [9]. These concepts are summarized in the following section, which discusses the system decoder model (SDM). This model is used to define the buffer and timing behavior of an idealized MPEG-4 terminal. Subsequently, the mechanism to packetize elementary streams and to convey timing information, i.e., the sync layer, is introduced.

4.1. The model

A system decoder model (SDM) is used to specify the behavior of a receiving MPEG-4 terminal in terms of a timing model and a buffer model. In contrast with MPEG-1 or MPEG-2, the SDM receives individual elementary streams from the delivery layer through the DMIF-Application Interface (DAI, see [8]). This design has been chosen because there is no single mandatory protocol stack below the DAI that accomplishes the

multiplexed delivery of elementary streams. Therefore, it is not possible to extend the model to cover multiplexing in a generic, yet meaningful way. MPEG-4 just puts requirements on the end-to-end delivery of data through the DAI, most prominently, a constant end-to-end delay. It is considered a task for the delivery layer below the DAI to guarantee this constant delay by suitable means. As a side effect, MPEG-4 neither specifies the additional end-to-end delay induced by multiplexing nor the associated delivery jitter. Such extensions of the SDM may, however, be developed in the context of specific MPEG-4 application scenarios.

The system decoder model is outlined in Fig. 16, and consists of the DMIF-Application Interface, a number of decoding buffers, decoders, composition memories and the compositor.

The core entity for the purposes of the SDM is the *access unit* (AU). Each elementary stream is partitioned in a sequence of such AUs. The semantic meaning of an AU is determined by individual media encoders and is not relevant either for the SDM or for the Systems perspective as such, with one important exception: the AU is the smallest entity to which timing information can be associated. The syntax of the sync layer (SL), described in the second part of this section, allows to encode both AU boundaries and the timing information associated to AUs.

The DAI supplies access units or parts thereof to the *decoding buffer* that stores the access units until their decoding time. At that point in time the SDM assumes instantaneous decoding of the access unit,

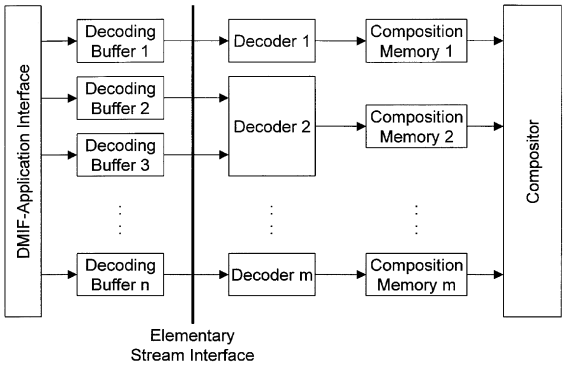


Fig. 16. System decoder model.

removal of the access unit data from the decoding buffer, and appearance of the decoded data corresponding to the access unit in the associated composition memory. With this model it is possible for the *encoding* terminal to know how much decoding buffer resources are available in the receiving terminal for a specific stream at any point in time.

The content of decoding buffers is consumed by *decoders*. In case of hierarchically encoded audio–visual objects, a decoder may be connected to multiple decoding buffers, as indicated with decoder 2 in Fig. 16. A decoder outputs the decoded data to one *composition memory*. The decoded data is grouped in *composition units*. The relation between access units and composition units need not be one-to-one but is assumed to be known for each specific decoder type. Each composition unit of decoded data is available for composition starting at an indicated composition time, either known implicitly or through an explicit composition time stamp, and ending at the composition time of the subsequent composition unit. The amount of composition memory required for specific audio–visual objects is not modeled by the SDM. A complete model with practical use would actually need to consider both the memory needed at the output of individual decoders and the memory needed as consequence of the transforms to be applied to this output according to the scene description.

The timing model postulates *object time bases* (OTB) to which the timing of elementary streams shall adhere. Such a time base is actually established at the sending side. Since in general it cannot be assumed that the sender and receiver are locked to the same clock, samples of the time base can be communicated to the receiver by means of *object clock reference* (OCR) time stamps. The receiver can then estimate, and therefore reconstruct, the speed of the sender clock by observing the arrival times of such OCRs.

In general, audio–visual objects whose elementary streams have different OTBs may be combined into a single presentation. A typical case would be a multi-point videoconference where it is not required that all sources run on the same time base. It is therefore not easily possible to synchronize contributions from different locations with each other in a strict sense, since there is no common, absolute,

clock. However, of course, it is still necessary to synchronize the audio and video from one location to achieve lip-synch. Therefore, the terminal has to recover the time bases from all contributing sources in order to present associated audio and video composition units at the right point in time.

In our networked world it is, of course, conceivable that even distributed sources are all locked to a single time base, that may be conveyed through the Network Time Protocol (NTP) [15] or through the Global Positioning System (GPS). In that case OCRs might still be used to indicate an arbitrary phase offset in the encoding of time stamps, however, a time base recovery is not necessary anymore. A single pair of OCR value and associated absolute time stamp (to be conveyed by means outside the MPEG-4 scope) would perfectly determine all subsequent time stamps.

Such *decoding time stamps* and *composition time stamps* convey the intended decoding time[1] of an access unit and the intended composition time of a composition unit, respectively. These times are expressed in terms of the OTB applicable for the given elementary stream. Many times the decoding and composition time will actually be the same. Different decoding and composition time for an access unit and the resulting composition unit(s) may be used with bi-directional predictive coding in video when transmission order and composition order of video object planes are reversed. Use of these two time stamps to offset a known non-instantaneous decoding duration is conceivable. This, however, is problematic since two different receiver implementations most likely would not have the same decoding delay.

Only a minimum of assumptions is made on the compositor for the purposes of defining the SDM. This includes that the compositor instantaneously samples the content of each composition memory of the MPEG-4 terminal. If the composition frame rate is high, visual composition units can actually be accessed more than once before the composition time of the subsequent composition unit is reached.

4.2. The sync layer

The sync layer is located between the compression layer that defines a binary representation specific to individual media types, and the delivery layer that merely conveys data packets from sender to receiver. It supplies information that is needed by both the compression and delivery layers. The sync layer provides a flexible syntax that encodes all relevant properties of access units, as they are defined by the compression layer[2], and allows the mapping of complete or partial access units into a delivery layer protocol.

The atomic entity of the sync layer is an *SL packet*. Such SL packets are exchanged with the delivery layer. The sequence of SL packets terminating in one decoding buffer of a decoder is termed an *SL-packetized stream*. This naming is somewhat awkward according to the common understanding of the notion *stream*, since actually it is not always possible (actually, not intended) to concatenate SL packets back to back and then obtain a parsable stream. In order to avoid a double encoding of the packet length information, the task of framing SL packets has rather been left to the delivery layer.

SL packets serve a double purpose. First, they allow fragmentation of access units in a content-agnostic way during adaptation to a delivery layer. Second, this fragmentation may as well be guided by the encoder. In that case SL packets are a convenient way to store the resulting SL-packetized stream including these fragmentation hints. This second purpose is motivated by real world scenarios where it is often beneficial if an encoder

[1] The model does not explicitly state what to do when an access unit has not arrived completely in the decoding buffer at its decoding time, since in that case the model is violated and there is no normative behavior. However, there are typically two ways to deal with such a situation: Either the AU is considered to be lost, i.e., in error, and the terminal continues from that stage, or the terminal "freezes" the internal clock (and, hence, possibly the presentation) in order to wait a little longer, assuming that it has enough memory to buffer all the other incoming data during that time. More flexible ways to indicate tolerances on decoding times already during content authoring are discussed for Version 2 of MPEG-4.

[2] To formalize this communication between compression layer and sync layer, an elementary stream interface (ESI) has been defined, which is, however, only of informative nature.

knows about some characteristics of the delivery layer, most prominently the size of the maximum transfer unit (MTU), i.e., the largest packet size that will be conveyed without fragmentation. If the compression layer is able to provide self-contained packets smaller than the MTU size, according to the application layer framing (ALF) principle [3], this usually leads to improved error resilience. In case of MPEG-4 video [5], such packets correspond to *video_packets*, while access units, of course, correspond to complete *video_object_planes*.

The sync layer syntax is flexible in that it can be configured individually for each elementary stream by means of the SLConfig descriptor, a mandatory component of the ES descriptor. This descriptor allows the selection of the length of many syntactic fields according to the requirements of this individual stream. For example, a very low bit-rate audio stream may require time stamps that consume few bits, while a higher bit-rate video stream may need very precise decoding time stamps. Under some conditions, time stamps can even be completely omitted if the access unit duration and the decoding time of the first access unit are known.

Each SL packet may contain two types of *sequence numbers* to discover lost SL packets and access units, respectively. AU sequence numbers also provide a simple mechanism to *repeat* access units, which is needed for scene description and object descriptor streams to enable random access in broadcast applications. Furthermore, clock reference time stamps and various flags to indicate presence of padding, start and end of access units, and the presence of a random access entry point are available. SL packets that contain the beginning of an access unit may have additional information about this AU, namely decoding and composition timestamp as well as the length of the AU or the instantaneous bit-rate of this stream.

In many applications multiple media encoders will actually use the same object time base, so that clock reference time stamps are only needed in one of the generated SL-packetized streams. Therefore, the SLConfig descriptor may simply provide a reference to another ES that conveys the clock reference for the current stream. It is even possible to create an elementary stream for the sole purpose of conveying clock references (with no media payload) in order to completely separate the two functionalities.

5. Elementary stream multiplex and delivery

As was stated earlier, Part 1 of MPEG-4 does not cover the domain of multiplexing and delivery of streaming data. A split approach has been followed instead. On the one side, Part 6 of MPEG-4 [7] specifies – among other things – the DAI: an abstract communication interface to arbitrary delivery technologies and protocols. On the other side, the actual definition of the necessary adaptations for the encapsulation of MPEG-4 SL-packetized elementary streams in concrete delivery layer protocols has to be pursued.

This section discusses approaches to the adaptation of MPEG-4 streams to such transport channels, termed payload format specifications, using the examples of the Internet and MPEG-2 transport environments. The last example discusses storage formats that are seen as conceptually very similar to transmission environments. All such adaptations eventually require standardization within the community that oversees the development or maintenance of a transport specification, e.g., the IETF in the case of the Internet and MPEG itself in the case of MPEG-2 transport.

As an exception to the general MPEG-4 approach, a simple multiplexing tool called *FlexMux* has been defined and is briefly introduced. It may be of value for reducing multiplexing overhead or delay in some environments. It may optionally be used in the transport protocol adaptation. Any delivery layer protocol into which SL-packetized streams are mapped has to provide framing of SL-packets, i.e., it has to implicitly or explicitly encode the packet size. Furthermore, in order to respond to quality of service requests by streams, a variety of different error protection mechanisms may be desirable in lossy delivery environments.

Furthermore, the ES_IDs that are provided through ES descriptors as labels for individual elementary streams need to be mapped to corresponding transport channel identifiers. The ES_IDs do not appear anywhere in the sync layer syntax, since this would be redundant with the transport channel

numbers that will appear in a multiplexing protocol anyway. The mapping is provided by conceptual *stream map tables* that will have a different structure and content depending on the actual delivery layer protocol.

5.1. The FlexMux Tool

The FlexMux is a multiplexer with a simple packet syntax, created for low bit-rate, low delay streams, as for example object descriptor, scene description, animation or speech streams. It is not regarded as a true transport level protocol but rather as a tool that should be used if the cost in terms of management load, overhead or delay to set up and use transport channels for each individual elementary stream would be too high. This may be the case if a presentation consists of dozens of audio–visual objects with a similar amount of corresponding elementary streams.

The two variations of the FlexMux packet structure are depicted in Fig. 17. In *simple mode* the header consists of an *index* that corresponds to the *FlexMux channel* (FMC), or stream number, and the packet length in bytes, both 8 bit, limiting the number of streams and payload size to 256 bytes each. The *MuxCode mode* is active for index values greater than or equal to 240. These values reference a payload template instead of an FMC. Payload templates, called *MuxCode table entries*, further reduce the multiplex overhead by describing how the payload of a single FlexMux packet is shared between multiple streams. This mode has an initial cost: each of the 16 possible templates needs to be conveyed before it can be used. MuxCode table entries should be conveyed by the same protocol that sets up this transport channel. If needed, MuxCode table entries may as well be changed dynamically. In order to maintain correct state and to discover transmission errors, a version number is used that must match between the current packet and the MuxCode table entry. The sequence of all FlexMux packets that are identified by a single FlexMux channel carry one SL-packetized stream. The sequence of all FlexMux packets that flow through one transport channel is termed *FlexMux stream*.

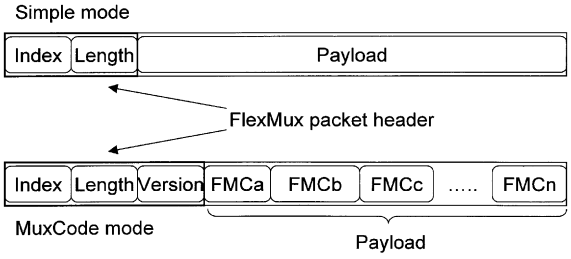


Fig. 17. FlexMux packet definition.

5.2. MPEG-4 content delivery on the Internet

There are several options for transport of real-time data on the Internet. The Real-time Transport Protocol [17] is evolving as one of the most used protocols. Issues of organizing real-time content into sessions and their control are addressed by accompanying protocols [18,13]. Adaptation of MPEG-4 to RTP is currently being discussed between MPEG and the IETF [2,11].

Other options like transport of MPEG-4 data on HTTP are not considered here, as HTTP is deemed to be a download protocol rather than a real-time transport protocol. Of course, a self-contained MPEG-4 file could be conveyed via HTTP. Similarly, direct transport of SL-packetized streams on UDP is not currently considered for standardization, even though this would be a perfectly viable scenario that has already been successfully demonstrated. However, the expectation is that RTP will be the focal point for the integration of MPEG-4 and non-MPEG-4 media types.

Both RTP and the MPEG-4 SL syntax provide ways to label a payload with a timestamp and enable loss detection through sequence numbers. While RTP has been designed for computational efficiency by respecting byte, word or even 32 bit boundaries for its syntactic elements, the SL syntax is focused more on flexibility and coding efficiency. Both expect the payload to be a semantically meaningful application data unit to facilitate stream-specific loss recovery according to the ALF principle [3]. Therefore, wherever possible, one elementary stream should be mapped into one RTP stream. In this case one SL packet will be conveyed per RTP packet. The redundancy between some

RTP header and SL header elements can be reduced to some extent as described in [2].

Since an MPEG-4 presentation may well consist of dozens of media streams, an additional mapping of multiple elementary streams into one RTP stream is beneficial. Both the overhead for individual RTP streams but even more so the management load induced by the parallel RTCP (RTP Control Protocol, also defined in [17]) streams and the increased number of lower level (IP) protocol packets suggest that the ALF principle should be set aside in this case by bundling multiple MPEG-4 elementary streams into one RTP session. IETF is currently in the process of developing a generic RTP multiplex that might serve this purpose. Alternatively, FlexMux may be used as an MPEG-4 specific solution to map a complete FlexMux stream in one RTP session.

While the payload format specification is making progress, the announcement and control of MPEG-4 content on the Internet has not yet been addressed with the IETF. The Session Description Protocol (SDP) [13] is a candidate for announcement purposes. Using SDP as a descriptor for an MPEG-4 presentation allows making use of other infrastructure that is currently being defined for content access. The Session Initiation Protocol [16] and Session Announcement Protocol [10] that are meant to announce multimedia events both use SDP descriptors. Similarly, SDP descriptors may be conveyed also via HTTP or email, using MIME types. Finally, RTSP [18] has a DESCRIBE command that conveys an SDP descriptor. RTSP is furthermore a candidate for session control and, more specifically, stream control purposes.

Given the properties of SDP, that is itself comparable to some extent to the object descriptors in MPEG-4, it is worth analyzing briefly whether SDP can completely take the role of the object descriptors in an Internet scenario. SDP allows to advertise media streams that belong to a multimedia session, including some level of descriptive information about the stream and an indication through which transport channel (UDP port) this media stream is conveyed. However, there is no explicit advertisement of hierarchical or alternative representations of a single audio–visual object and, of course, there is no predefined way to associate specific streams to such audio–visual objects at all. There is also no means to associate intellectual property rights information or object content information to individual audio–visual objects nor can the number of streams in the session be dynamically modified.

Of course, it is conceivable to extend SDP in this direction but, assuming that the Internet will not be the only medium on which MPEG-4 content will be released, it seems preferable to keep the MPEG-defined object descriptors to describe the elements of an MPEG-4 session and to use SDP at the delivery layer to simply point to this session as a whole. In that case, SDP can be used to convey the initial object descriptor as well as the stream map table. The syntax of the stream map table remains to be defined.

5.3. MPEG-4 content delivery on MPEG-2 Systems

MPEG-2 Transport Streams (TS) and Program Streams (PS) provide the transport encapsulation for real-time digital broadcast data and DVD, respectively. The specification includes also all the necessary descriptors (Program Specific Information, PSI) to identify the services communicated within a TS or PS. The Packetized Elementary Stream (PES) syntax serves as a common denominator to encapsulate content to be conveyed both in TS or PS.

The draft specification for MPEG-4 content delivery on MPEG-2 Systems [12] defines the encapsulation of both SL-packetized streams and FlexMux streams in PES packets. In the first case one SL-packetized stream is mapped to one "channel" of the MPEG-2 multiplex. Such a channel is identified by a PID (packet identifier) in case of TS and by a stream id in case of PS. Each SL packet is mapped to one PES packet. The redundancy between PES and SL packet header is reduced by conveying the information (such as time stamps) only in the PES header and removing the respective items from the SL packet header.

Despite the redundancy reduction, this mode has a rather high multiplex overhead. Therefore, it is possible as well to encapsulate an integer number of FlexMux packets in a PES packet. Here, several

SL-packetized streams are mapped to one MPEG-2 PID or stream id. In this mode the PES header is not used at all, i.e., synchronization is done with the time stamp information conveyed in the SL packet headers. The multiplex overhead is two byte per FlexMux packet, or even lower, if the MuxCode mode is used, compared to a minimum of 6 byte per SL packet in the first case.

Additionally, a set of MPEG-4-related descriptors is being defined in extension of MPEG-2's PSI. The initial object descriptor for an MPEG-4 session will be carried in a Program Map Table (PMT) entry. Additional descriptors, corresponding to the stream map table, establish the correspondence between the ES_IDs that identify elementary streams within the session and the transport channel consisting of either a PID or stream id and, optionally, a FlexMux channel number.

5.4. Storage of MPEG-4 content

The storage of MPEG-4 presentations seems to be an issue that is not directly related to the previously described mappings to transport formats. Conceptually, however, it is not much different. MPEG-4 presentations may consist of a large number of elementary streams and, as before, it has to be decided whether each of the streams is to be stored in a separate file or whether streams share one file, potentially through the use of FlexMux. It is not intended to directly use the file format for transmission of MPEG-4 streams. A translation to a transmission format that is appropriate for the target transport network should be done.

An MPEG-4 file format is currently being developed, starting from Apple's QuickTime™ file format that has been chosen out of a number of proposals (see, for example, [1]) as a starting point. It is currently being adapted to the needs of MPEG-4 and will be included in the second version of the MPEG-4 standard that is expected to be approved at the end of 1999.

6. MPEG-4 content access procedure walkthrough

To conclude this paper, we now outline the MPEG-4 content access procedure using the building blocks previously described. This access may occur in different ways, depending on the application context. The details differentiating the different access procedures are confined to the delivery layer, below the DAI, and are not further detailed.

It is assumed that the MPEG-4 receiver application is already active. How it has been activated is out of the scope of the MPEG-4 specifications. It is also assumed that the URL corresponding to a service has been made available to it, by means that are also out of the scope of the MPEG-4 specifications (e.g., a link on the Web).

Given these preconditions, the following steps shall take place (bearing in mind that the usage of the DAI is instrumental to the description of the walkthrough, but is not mandatory in a compliant implementation):

1. The receiver requests the service by passing a URL to the DAI; as a result a service session is established and the initial object descriptor is returned.
2. The ES descriptors in this initial OD identify the primary OD stream(s) and scene description stream(s) for this service.
3. The receiver selects the ES_IDs (or URLs, if any) of those OD and scene description streams that it requires.
4. The receiver requests delivery of the streams with these ES_IDs (or URLs) through the DAI; handles to the channels carrying the respective streams are returned.
5. The receiver requests to play the data.
6. The sender starts delivering the stream (which in a broadcast scenario may actually mean simply: starts reading data from the network), and the receiver accesses data through the DAI.
7. The receiver parses arriving BIFS and OD data.
8. The receiver selects ES_IDs (or URLs) of those media streams that it requires.
9. continue at step 4.

As can be seen, the MPEG-4 content access requires multiple requests through the DAI. In some scenarios this might lead to the definition of 'well-known' ES_IDs, so that some streams can be acquired without having to go through the entire procedure. In case of broadcast applications, the requests through the DAI are of course implemented

on the delivery layer by means that do not require a backchannel. Furthermore, the process deliberately ignores questions of delays induced by establishing access to various, potentially distributed resources. This has to be taken care of by making all relevant object descriptors available sufficiently ahead of the time at which the connection to the ESs is actually needed.

7. Conclusion

The elementary stream management in the MPEG-4 standard has been described. Its basic building blocks are the object description framework and the means to synchronize elementary streams. Object descriptors are the means to link audio–visual objects in the scene description to the elementary streams that carry their encoded information. Object descriptors contain a set of various other descriptors, in order to fully describe the properties of the streams and their associations. Use of time-stamped object descriptor commands allows for the creation of very dynamic content, in which objects – and thus streams – may be added or removed on-the-fly.

Timing and synchronization features have been considerably modified compared with MPEG-2. The traditional combination of clock references and time stamps allow for object clock recovery and audio–video synchronization. Synchronization based on stream start times and known duration of individual access units has been added. It is possible to compose content originating from multiple sources into one MPEG-4 presentation. Perfect synchronization is possible if such sources are locked to a globally available time base, such as GPS.

The current state of the specifications for Internet-based and MPEG-2 delivery has been outlined, as they are expected to be the dominant means of bringing MPEG-4 content to users. All specifications for MPEG-4 delivery have to be addressed within the standardization bodies responsible for the transport layers concerned. There is already considerable progress in these areas, and we expect that complete solutions will be defined soon. A very important delivery layer adaptation, the MPEG-4 file format will become available as part of the second version of the standard that is anticipated for the end of 1999.

Acknowledgements

This paper reflects the work of many people around the world who dedicated their creative abilities to the success of MPEG-4 Systems. We would like to thank all of them for their active involvement and dedication to MPEG-4's goals.

References

[1] A. Basso, H. Kalva, A. Puri, A. Eleftheriadis, R.L. Schmidt, The MPEG-4 file format: An advanced, multifunctional standard for new generation multimedia content, IEEE Trans. Circuits Systems Video Technol. (Special Issue on Synthetic Natural Hybrid Coding) (1999) to appear.

[2] R. Civanlar, V. Balabanian, A. Basso, S. Casner, C. Herpel, C. Perkins, RTP payload format for MPEG-4 Streams, Internet-Draft draft-ietf-avt-rtp-mpeg4-01.txt, in preparation.

[3] D.D. Clark, D.L. Tennenhouse, Architectural considerations for a new generation of protocols, in: Proceedings, ACM SIGCOMM, Philadelphia, Pennsylvania, September 1990, pp. 200–208.

[4] Coding of Audio–Visual Objects: Systems, ISO/IEC 14496-1 Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2501, November 1998.

[5] Coding of Audio–Visual Objects: Visual, ISO/IEC 14496-2 Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2502, November 1998.

[6] Coding of Audio–Visual Objects: Audio, ISO/IEC 14496-3 Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2503, November 1998.

[7] Coding of Audio–Visual Objects: Delivery Multimedia Integration Framework, ISO/IEC 14496-6 Final Draft International Standard, ISO/IEC JTC1/SC29/WG11 N2506, November 1998.

[8] G. Franceschini, The delivery layer in MPEG-4, Signal Processing: Image Communication 15 (2000) 347–363.

[9] Genetic Coding Of Moving Pictures and Associated Audio: Systems, ISO/IEC 13818-1, 1994.

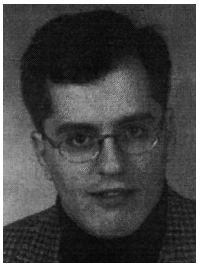
[10] M. Handley, V. Jacobson, SAP: Session announcement protocol, Internet-Draft draft-ietf-mmusic-sap-00.txt, in preparation.

[11] C. Herpel (Ed.), Architectural Considerations for Carriage of MPEG-4 over IP Network, ISO/IEC JTC1/SC29/WG11 N2615, December 1998.

[12] ISO/IEC 13818-1/Final Proposed Draft Amendment 7, ISO/IEC JTC1/SC29/WG11 N2664, March 1999.

[13] V. Jacobson, M. Handley, SDP: Session Description Protocol, RFC 2327, 1998.
[14] R. Koenen, Profiles and levels in MPEG-4: Approach and overview, Signal Processing: Image Communication 15 (2000) 463–478.
[15] D. Mills, Network Time Protocol (NTP) Version 3.0, RFC 1305, March 1992.
[16] E. Schooler, H. Schulzrinne, M. Handley, SIP: Session initiation protocol, Internet-Draft draft-ietf-mmusic-sip-11.txt in preparation.
[17] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson, RTP: A Transport Protocol for Real-Time Applications, RFC 1889, 1996.
[18] H. Schulzrinne, A. Rao, R. Lanphier, Real Time Streaming Protocol (RTSP), RFC 2326, 1998.
[19] J. Signès, Y. Fisher, A. Eleftheriadis, MPEG-4's binary format for scene description, Signal Processing: Image Communication 15 (2000) 321–345.

Carsten Herpel was born in Cologne, Germany in 1962. He received his diploma in communication engineering from RWTH Aachen, Germany, in 1988. He then joined Thomson Multimedia's research facility in Hannover, Germany, and developed video coding algorithms, with a focus on scalable coding tools. His research interests have then moved to multimedia systems with a current focus on MPEG-4. Mr. Herpel serves as a co-editor to the MPEG-4 Systems specification.



Alexandros Eleftheriadis was born in Athens, Greece, in 1967, and received his Ph.D. in Electrical Engineering from Columbia University in 1995. Since 1995 he has been an Assistant Professor at Columbia, where he is leading a research team working on media representation with emphasis on multimedia software, video signal processing and compression, video communication systems and the mathematical fundamentals of compression. Dr. Eleftheriadis is a member of the ANSI NCITS L3.1 Committee and the ISO-IEC JTC1/SC29/WG11 (MPEG) Group, where he serves as the Editor of the MPEG-4 Systems specification. His awards include an NSF CAREER Award.