Journal of VLSI Signal Processing 27, 55–67, 2001
© 2001 Kluwer Academic Publishers. Manufactured in The Netherlands.

MPEG-4 Systems: Architecting Object-Based Audio-Visual Content

ALEXANDROS ELEFTHERIADIS
Department of Electrical Engineering, Columbia University, New York, NY 10027, USA

Received July 20, 1999

Abstract. We describe the architecture and key features of the MPEG-4 Systems specification, as well as the encoding methodology of its various components: scene description and BIFS, animation streams, object descriptors, object content information, as well as delivery and multiplexing. We also describe the MPEG-4 reference software as well as our own prototype software for MPEG-4 authoring, streaming, and playback. Finally, we briefly compare MPEG-4 Systems with a number of currently available alternative standards and commercial solutions.

Keywords: MPEG-4 systems, object-based representation, digital audio and video, interactive multimedia

1. Introduction

MPEG-4 [1–3] is a standardization effort under the auspices of ISO/IEC [4] being developed by MPEG (Moving Picture Experts Group) [5], the committee that also developed the Emmy Award-winning standards known as MPEG-1 and MPEG-2 [6]. MPEG-1 and MPEG-2 have evolved to be the dominant formats for digital audio and video compression and distribution, in both Internet as well as professional applications. For example, MPEG-1 and 2 Audio Layer III (known as MP3) is the dominant format for music distribution on the Internet today, whereas MPEG-2 video is the basis for digital broadcast TV, DVDs, as well as satellite TV (DBS systems such as DirecTV etc.).

Following the success of these standards, the MPEG group embarked on an investigation of the technology direction required by the digital audio and video content industry. At the beginning, significant focus was placed on the coding aspects, and in particular on very low bit rate coding (below 64 Kbps). It was soon realized, however, that coding improvements could not be as dramatic as one would hope for. New requirements for audiovisual applications were also put forward—interactivity, in particular, became a dominant theme in the group's work. Interestingly enough, at the time the foundation of MPEG-4 was being put together (1994-95), the Internet and the Web started to gain significant momentum in the United States and abroad. This affected MPEG-4 in several ways, although the focus remained on audiovisual applications. A brief history of MPEG-4, including a timeline of its development, is provided in [7].

MPEG-4 is building on the proven success of three fields: digital television, interactive graphics applications (synthetic content) and the World Wide Web (distribution of and access to content) and intends to provide the standardized technological framework enabling the integration of the production, distribution and content access paradigms of the three fields.

MPEG-4 addresses the "generic coding of audio-visual objects." In contrast to all other existing audio or video representation standards, MPEG-4 adopts an object-based approach for content description: the content is assumed to be constructed out of individual and independent entities called objects, which are separately encoded. These objects include, for example, arbitrarily shaped natural video, graphics, natural or synthetic audio, face or body animation etc. MPEG-4 has defined a number of representation tools to address the coding needs of a large variety of media. These tools extend much beyond the ones used in MPEG-1 and MPEG-2, which only addressed natural video and audio at combined bit rates of 1–20 Mbps. For example, MPEG-4 includes mesh coding tools, scalable coding of still images using zero-tree wavelets, face animation parameter coding, etc.

Coding of such objects, however, is only the first step into constructing a complete multimedia scene. Additional information is needed in order to: 1) describe how these objects should be placed in space and time, 2) how they may interact with each other and the end-user, 3) how to multiplex all this information into one or more streams for delivery over a variety of networks, and 4) ensure proper synchronization among the various streams. This information is the realm of the MPEG-4 Systems specification (Part 1 of the MPEG-4 standard) [1], which is also responsible for the overall architectural definition of MPEG-4.

Traditional audiovisual coding standards are based on the use of one stream for video data coupled with one ore more streams for the associated audio. These streams are multiplexed together into a single stream that also carries timing information that allows a receiver to reconstruct the sender's clock (this information is important in broadcast environments where no other means of sender/receiver flow control are available). Furthermore, the streams contain buffer control information that enable management of the receiver's buffers (e.g., the VBV model in MPEG-2 [6]). Timing recovery and multiplexing are the traditional domains of the 'systems' layer.

The object-based nature of MPEG-4 required a reengineering of this simple architecture to accommodate the many new features targeted by the specification. In particular, content representation is separated into three major entities: object descriptors, scene description, and coded audiovisual data. A fourth category, object content information, can optionally be used as well and is described in more detail later on.

MPEG-4 Systems also uses a syntactic description language (Flavor [8, 9]) to describe the bitstream syntax, in contrast to the ad-hoc mechanisms used in prior MPEG and other specifications. Several of the coding processes use Flavor's particular object-based features for maximizing flexibility and conciseness. Flavor provides for automatic code (C++ and Java) generation from the description of the bitstream syntax, thus significantly simplifying the task of codec development [9].

In the following, we first provide an architectural overview of MPEG-4. We then describe each of its components, starting from the traditional domain of synchronization and buffer control, and moving to the novel concepts of object descriptors, scene description, delivery and multiplexing, and object content information.

More detailed information about MPEG-4 can be found in [7, 10, 11], and of course the text of the standard itself [1]. A high-level overview is available in [12]. We should point out that Version 1 of the specification is already final as of January 1999, while a number of new features are scheduled to be added in January 2000 in the form of Version 2. The new version is an incremental update of Version 1 (in the form of ammendments) and includes features which were not fully developed and/or tested by the time Version 1 was finalized. A description of these features is provided at the end of this paper.

2. The Architecture of MPEG-4

Figure 1 shows an architectural overview of MPEG-4. There are three fundamental components comprising MPEG-4 content: object descriptors, scene description, and individual audiovisual object data. The latter correspond to traditional natural digital audio and video streams, but can also include still images as well as synthetic content. MPEG-4 has defined a number of new tools to encode a variety of media types. Most notable are arbitrarily-shaped digital video, still images, and synthetic (structured) audio. Parts 2 (Visual) and 3 (Audio) of the MPEG-4 specifications detail the bitstream formats and the decoding processes for these tools.

Each entity is conveyed in its own *elementary stream*. These streams may have been multiplexed into one or more streams for transport over a particular network (e.g., a single TCP stream or a set of RTP streams for IP-based transport, or a single MPEG-2 Transport Stream); if so, we assume here that they have been already demultiplexed. MPEG-4 is transport agnostic and does not require or define any particular transport facility. As we discuss later on, a simple multiplexer is defined for delivery systems that lack one (e.g., GSM data channels) but its use is entirely optional.

Object descriptors are used to identify a set of elementary streams as a single audio-visual object. These descriptors are associated with identifiers that allow them to be referenced from the scene description, and fully characterize the elementary stream(s) that comprise the object at hand. More than one stream may be associated with a given audiovisual object when

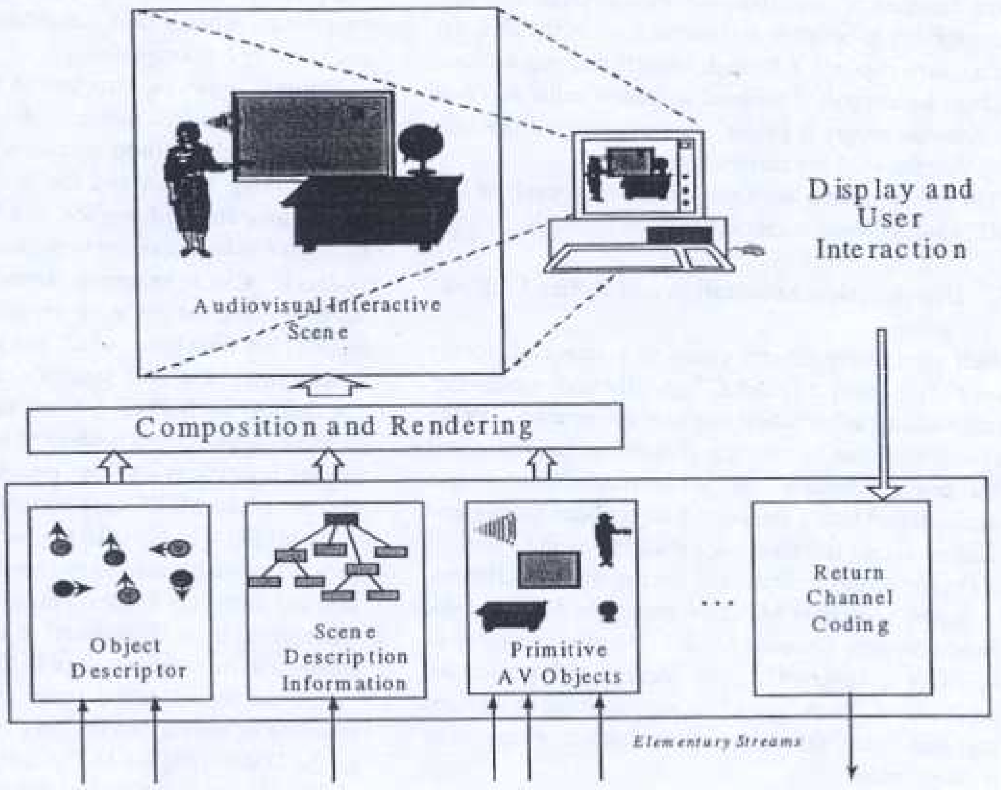

Audiovisual Interactive Scene
Display and User Interaction
Composition and Rendering
Object Descriptor
Scene Description Information
Primitive AV Objects
...
Return Channel Coding
Elementary Streams

Figure 1. Architectural overview of MPEG-4.

scalability is used, or when multi-channel or multi-language audio is available.

The scene description defines the spatio-temporal behavior of these objects and incorporates potential user interaction. The scene is composed and rendered on the system's presentation devices according to the scene description.

It is important to note that this architecure represents a significant departure from traditional content creation and distribution models. Typically, content is acquired and then edited/composed at the production studio, followed by conversion to a distribution format (e.g., MPEG-2) for transmission to end-users. In MPEG-4, composition of the objects comprising a scene occurs at the user's terminal (be it a stand-alone device or a program running on a general-purpose computer).

This architecture is not very different from the one used on the Web: here an HTML page acts as the scene description, which also contains the textual information (the primary medium). All other forms of content, however, are referred to from the HTML file via URLs, and are retrieved and composed on the page by the browser. There are, of course, considerable differences between MPEG-4 and the Web, since the former has to contend with issues of timing and synchronization and compression, among others.

The fact that composition occurs at the receiving terminal implies that the latter must have considerable computational power. However, it also simplifies tremendously the editing process. As an example, tasks such as logo insertion become completely trivial with MPEG-4, since they only involve the addition of simple scene description commands and transmission of the logo in its own stream.

More importantly, by transmitting a scene as a decomposed set of objects that will be composed and rendered at the receiver, we now have the capability to individually identify each object and assign events and event handlers to them. Also, the fact that the scene structure survives the process of distribution has a profound impact on applications that involve text or content-based queries. Since the scene structure is available at the receiver, it can be used for filtering as well as retrieval applications. MPEG-4 further facilitates these tasks by providing *object content information*, i.e., tags that provide descriptive information about the content.

Interaction with the sender is also possible via a return channel, if available. We should point out that interactivity is possible and useful even in the absence of a return channel. Although trasactional applications cannot be supported without it, added value services (electronic program guides, movie summaries including visuals, etc.) are certainly possible.

In the following sections we describe each of the MPEG-4 Systems components in more detail.

3. Delivery, Synchronization and Buffer Control

Each elementary stream contains a series of *access units* of the entity it conveys. Typically, each access unit encapsulates information that needs to be associated to a particular time instant (e.g., for video, an access unit is a "frame"). Each access unit (or fragment thereof) is encapsulated into a structure that contains timing and random access information, called the *Sync Layer*.

For the proper definition of the timing and buffer behavior of compliant MPEG-4 terminals, MPEG-4 defines a System Decoder Model. The timing model of MPEG-4 is designed to allow maximum flexibility for applications. Both "push" (e.g., broadcast or streaming) and "pull" (receiver-driven) modes of operation are supported.

For applications that require open-loop flow control timing recovery can be effected through the use of object clock references (similar to program clock references of MPEG-2). Each object is assumed to have its own time base, that must be mapped to the system time base. Each object decoder is further associated with particular buffer resources which are managed via decoding and composition time stamps. Notice that MPEG-4 does not refer to presentation timestamps, because composition and rendering are not defined by the standard.

The presence and resolution of all such timing information is entirely optional. It is also possible to specify a particular rate in lieu of time stamps, thus eliminating the need for lengthy timing information. The configuration of the sync layer packet header is part of the elementary stream descriptors which are described in the next section.

Due to its object-based nature, an MPEG-4 terminal may have to operate several media decoders. Each of these decoders has its own dedicated decoding buffer. Also, since composition must take place after decoding, the output of a decoder is sent to a *composition buffer*. The size of this buffer is not normatively specified, and is assumed to be large enough to hold at least one presentation unit. The contents of this buffer are overwritten when a new presentation unit is decoded and is ready for composition.

Figure 2 shows the structure of the decoders within an MPEG-4 terminal, including decoding and composition buffering. The (non-normative) interface between the decoding buffers and the decoders is called the *Elementary Stream Interface*, and defines the information that has to be carried from these buffers to the decoders in order to ensure the latter's proper operation. As mentioned earlier, each object (and thus decoder) has its own time base, which has to be mapped to the system time base. The system's clock parameters are not defined by MPEG-4, since they are application-dependent. Individual *profiles* or *levels*, however, may impose limitations on these parameters as they did in the case of the MPEG-2 specification.

As MPEG-4 is designed to operate under a large variety of network transport mechanisms, it does not provide any particular transport facilities. Transport layers are referred to as 'TransMux' in the MPEG-4 specification, and only the interface to this layer is defined.

In order to isolate the design of MPEG-4 from the specifics of the various delivery systems, the concepts of the DMIF (Digital Media Integration Framework) and DMIF Application Interface (DAI) were defined (see Fig. 2). The DAI defines the process of exchanging information between the terminal and the delivery layer in a conceptual way, using a number of primitives. It should be pointed out that this interface is non-normative; actual MPEG-4 terminal implementations do not need to expose such interface. For networks that do not provide appropriate multiplexing facilities, MPEG-4 defines a simple optional tool called 'FlexMux' that is particularly suitable for low-delay applications.

The FlexMux originated from the H.223 Annex A multiplexer, and uses very small packets (up to 256 bytes). It provides for simple packetization with a small header that identifies the channel number and the packet length. It also supports a more complex mode, where different bytes of a packet can be assigned to different channels (so-called 'muxcode' mode). Configuration information must be provided to the receiver in order to bind different slots to the various channels. In both

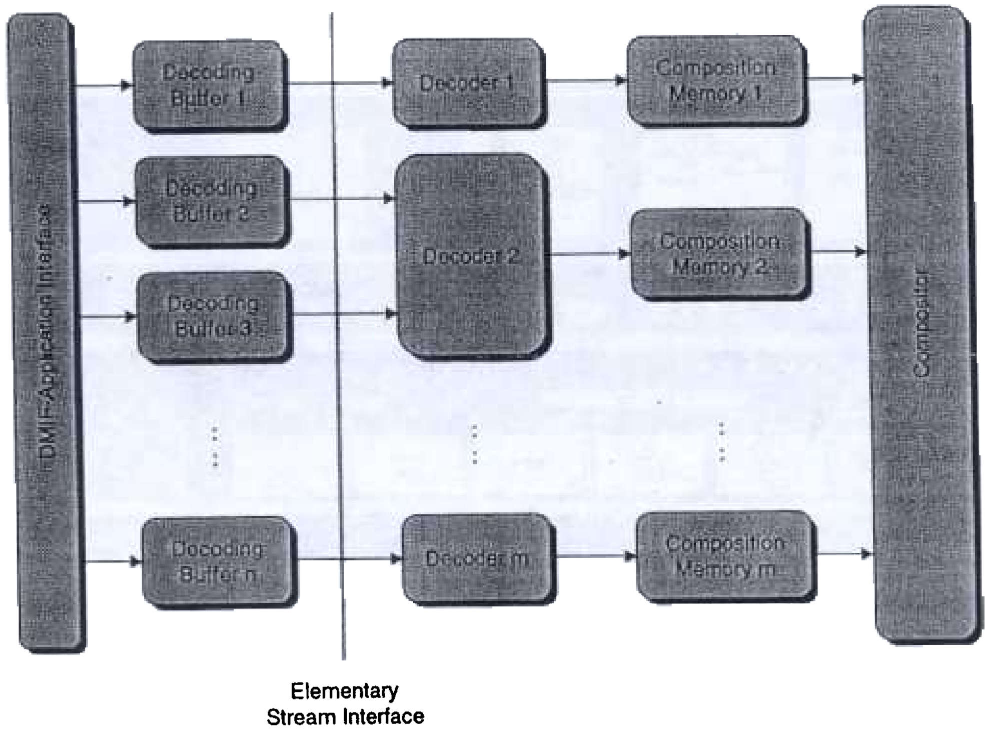


Figure 2. System decoder model.

modes, a stream map table must be provided out of band in order to define the contents of each channel.

Figure 3 depicts the layering of the sync layer, FlexMux, and TransMux in an MPEG-4 system. Notice that the use of the FlexMux is optional, and that a large variety of transport facilities can be used (the list shown in the figure is *not* intended to be exhaustive). The entire structure below the Sync Layer is referred to as the "delivery system." The term is meant to encompass networked as well as mass storage distribution facilities. As we discuss later on, MPEG for the first time is defining (for Version 2) a file format, called MP4, to be used for content interchange purposes.

As a result of its network independence, MPEG-4 says very little about how content should be transported over specific delivery systems such as the Internet. The intention is that the specifics have to be defined by the bodies responsible for the design and evolution of these systems. For example, collaborative work is already underway with IETF's AVT group in order to define appopriate facilities for the transport of MPEG-4 content over RTP. Similarly, work is underway within MPEG to define mechanisms for transport of MPEG-4 content over MPEG-2 Transport Streams.

4. Object Descriptors

Object descriptors are a fundamentally new concept in MPEG, and are a key data structure for the operation of an MPEG-4 terminal. Object descriptors have two primary purposes. First, they are used to define the set of elementary streams that carry an object's data, as well as the properties of these streams (e.g., their Sync Layer configuration, the format of the data, whether it is audio, video, scene description, object descriptor stream, etc.). Second, they are assigned unique (within a session) identifiers so that they can be referenced by the scene description. By completely separating the scene description from the audiovisual data the process of content creation and editing is considerably simplified.

Object descriptors are carried in their own elementary stream(s), packaged into Sync Layer packets, with timestamping information if so desired. In essence, object descriptors announce to the receiving terminal the different types of objects that are available in the current session and also provide all of the configuration information required for their decoding. Given the object descriptors, the only information missing from the terminal is when and where to compose each

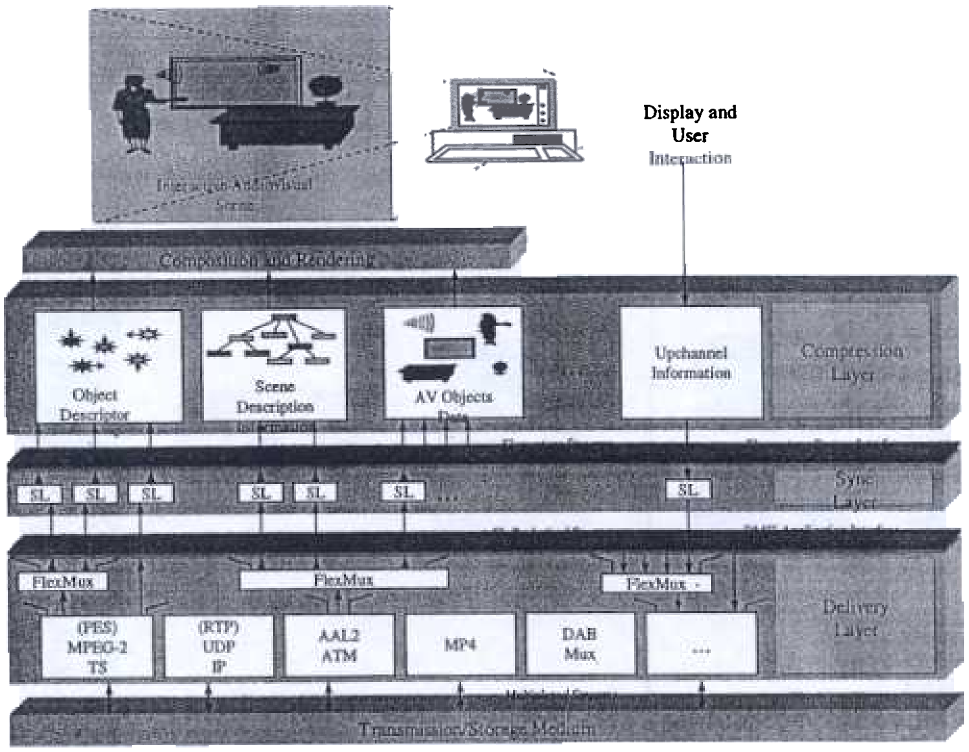


Figure 3. Layering of an MPEG-4 system

audiovisual object. This information is provided (as discussed later on) by the scene description.

Note that more than one stream may be associated to an object due to scalable coding or due to the use of multichannel audio coding. Also, an elementary stream may be associated to more than one object as well.

Each descriptor is associated with a 10-bit identifier that must be unique within an MPEG-4 session. Object descriptors can be inserted, deleted, or updated at any time using time stamped object descriptor commands conveyed in the same elementary stream.

Each descriptor contains a set of mandatory and optional sub-descriptors. Most important is the set of elementary stream descriptors, which identifies the elementary streams that are part of the object at hand. These descriptors not only identify the particular streams, but also provide information about the configuration of their sync layer and initialization information required by the corresponding decoder. Stream identification, similarly to object descriptors, is performed via a 5-bit identifier. This identifier is resolved to a particular elementary stream using a stream map table that is specific to the transport facility used, and hence is not defined by MPEG. Optional sub-descriptors can provide information about the Quality of Service (QoS) required by the stream, language, or intellectual property information.

A special object descriptor is required to bootstrap an MPEG-4 session. This descriptor is called the *initial object descriptor* and contains the elementary stream descriptors for the object descriptor and the scene description streams. The initial descriptor is assumed to be delivered out-of-band, via application-specific means (e.g., a URL).

Figure 4 shows how object descriptors are used to obtain access to content [13]. The initial object descriptor contains elementary stream descriptors that provide the location (stream ID) and configuration of the scene description and object descriptor streams. With this information, the receiving terminal can access the relevant streams, depacketize them, and decode their contents. Each object descriptor presumably causes the instantiation of a decoder for the respective media type. The terminal then has to process the scene description information, and apply the composition information on the composition buffers of the various decoders.

We should note that both object descriptors as well as elementary stream descriptors can include URLs.

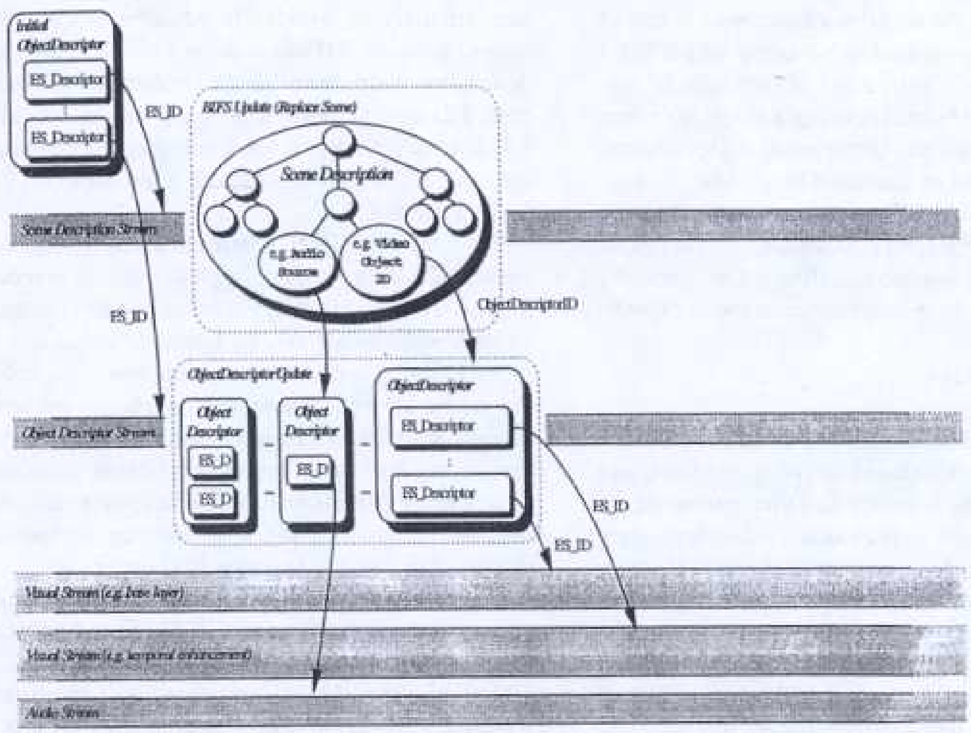


Figure 4. The object descriptor framework.

This allows the construction of an MPEG-4 session in a distributed fashion, collecting objects or streams from more than one location. The presence of HTTP URLs implies that the system decoder model can no longer be applied, due to the best-effort nature of the Internet. As a result, MPEG-4 does not provide any details on how a terminal should behave when URLs are utilized.

Figure 5 depicts the contents of an object descriptor as well as the elementary stream descriptor. As we see, the former is either a URL, or an identifier followed by a series of elementary stream descriptors, Object Content Information (OCI) descriptors (described later on), and Intellectual Property Management and Protection (IPMP) descriptors. The latter are used as hooks to enable the integration of copyright and general content protection mechanisms (defined outside of MPEG, by service providers or industry organizations).

The elementary stream descriptor contains information pertaining to decoder configuration (initialization parameters etc.) and sync layer configuration (packet header options and so on). Additional optional information can also include Intellectual Property Information (ISIP—tags for the identification of copyright holder, ISBN and related numbers, etc.), IPMP information (similar to an object descriptor), a language descriptor, as well as QoS information. The latter identifies the QoS requirements of the particular stream that this descriptor refers to.

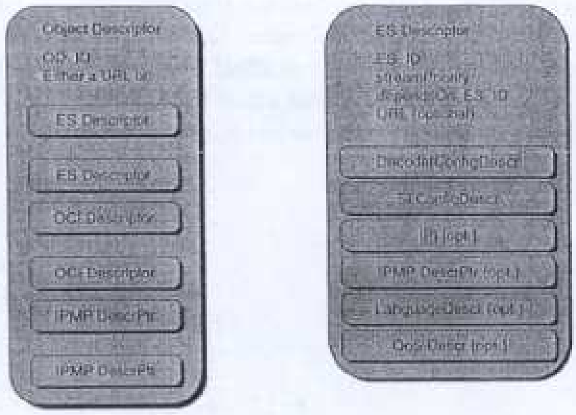


Figure 5. Object descriptor and elementary stream descriptor structures.

The object descriptor framework has been defined in an extensible manner, so that new descriptors can be added without affecting interoperation with older terminals. This is very important as descriptors are a very convenient mechanism to introduce meta-data about the content, and it would be difficult if not impossible to predict what types of information may be needed (or available) in the future.

One drawback of the descriptor framework is that an initial descriptor is required to bootstrap an MPEG-4 session. Due to the involvement of the delivery system, this has resulted in not defining a single bootstrap mechanism (since delivery of the initial object descriptor will be performed as mandated by the body that supervises the particular delivery system used). Specific instances are MP4 files, RTP, DSM-CC, and MPEG-2 TS; mechanisms for session initialization are currently under design for all these very important environments.

Scene Description

The scene description information describes how the various objects are positioned in space and time, and also defines dynamic behavior and user interaction. It does not directly refer to particular media elementary streams, but rather refers to object descriptors via their identifier. This completely decouples the scene description from the specifics of the encoding of particular objects. This allows, for example, changing the contents/encoding of a video object without any need of modification of the scene description itself.

The scene description is heavily based on the VRML-97 [14] specification and there is a concerted effort to completely align the two specifications. The scene is represented as a tree of nodes. Each node has an associated number of fields that affect its behavior. Leaf nodes are media objects, with a field that refers to either object descriptors (via the object descriptor identifier) or a URL. In the latter case, MPEG-4's System Decoder Model does not apply.

Intermediate nodes perform grouping, translation, etc., similarly to VRML. In addition to the VRML nodes, however, MPEG-4 defines a set of 2-D scene description nodes to enable the implementation of low-cost, 2-D only systems. It is also possible to combine 2-D scenes with 3-D scenes, using appropriate *layering* nodes. The overall structure of an MPEG-4 scene is shown in Fig. 6.

In contrast to VRML where the scene description is static, MPEG-4 provides complete freedom in modifying the scene description via scene update commands. In particular, nodes can be inserted, updated, or removed; similar operations can be applied to the fields of the nodes as well. These commands are performed according to the timing information of the sync layer that is used to carry scene description information as any other MPEG-4 data. Scene update commands become effective at the time directly or indirectly conveyed in the sync layer packet header.

The binary encoding of scene descriptions is called Binary Format for Scenes (BIFS). Encoding is performed in a depth-first fashion. Each of the almost 100 nodes defined in the specification is identified by a number. The fields of each node assume default values, unless explicitly overridden in the bitstream. Each field is itself identified by an ordinal number; the type of the field (e.g., an integer, a float, or another node) is implicitly obtained from that number and is defined in (extensive) node coding tables. To maximize efficiency, the fact that only particular nodes can be children of a given node is used to introduce "context" in the coding

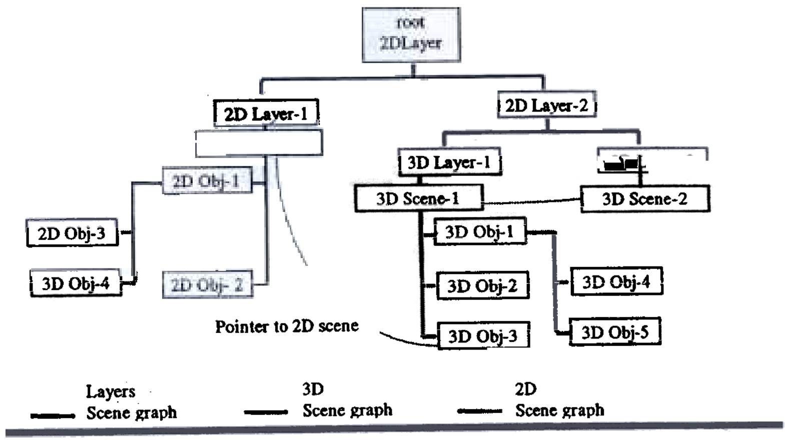


Figure 6. A typical MPEG-4 scene structure.

process. Depending on that context, called the *node data type*, a different number of bits as well as different node codes are used to identify the various nodes. For improved coding efficiency, quantization facilities are also provided for field values and are introduced by special 'QuantizationParameter' nodes. Nodes that may be updated are associated with a node identifier (similar to object descriptors), so that later commands can refer to them.

To facilitate dynamic scene behavior and user interaction, VRML's concept of *routes* is also used in MPEG-4. Some fields of a node are categorized as event sources or sinks. Routes are then used to link a source to a sink, allowing the implementation of simple state transition triggers. Routes are coded after the scene description tree is coded, and use the node identifier of the source and sink nodes to identify the event source and sink fields.

The BIFS mechanism provides a quite efficient compression tool for scene descriptions. A simple scene emulating traditional MPEG-2 video takes only 4 bytes. In some cases, however, the scene update mechanism via BIFS access units may present a significant overhead. This is particularly the case for low bit rate applications where simple field value modifications may be desired. For these cases, a light-weight update mechanism is provided via BIFS *animation streams*. These can only provide updates of one or more fields of a particular node in a way that avoids the node- and field-related overhead of regular BIFS updates, and achieves further compression efficiency using predictive as well as arithmetic coding.

To facilitate complex state behavior, scripting capabilities are provided using ECMAScript (the official name of JavaScript). Scripts are carried within the scene description using a 'Script' node, which contains an encoded version of the actual script. The inclusion of scripting provides a powerful mechanism for application development. Version 2 will further enhance this capability through the inclusion of Java support in MPEG-4 terminals, as described below.

6. Object Content Information

Object Content Information (OCI) is the mechanism with which ancillary identification information can be associated with an object. Such information includes a cataloguing number and authority identification (e.g., ISBN numbers), keywords in any language, ratings, author information and date of creation, and so on.

OCI can be carried in an object descriptor, or be assigned its own elementary stream. In the latter case, the information can change over time to reflect potentially changing attributes for a given object.

The use of OCI is crucial for the implementation of content filtering as well as content retrieval applications. The scene description itself provides significant information regarding the scene *structure*. Coupled with *semantic* information that can be carried within OCI (in the form of textual and other attributes), the receiver or query engine has a wealth of information at its disposal to make informed content seletion decisions.

The concept of OCI will be further expanded upon in the next MPEG project (MPEG-7), which focuses exclusively on content description interfaces.

7. MPEG-4 Version 2

While MPEG-4 contains a significant number of features, several iterms were not fully developed or tested in time to be included in the final version of MPEG-4. As a result, a comprehensive list of ammendments is being developed, commonly referred to as Version 2. We should stress that this represents an incremental upgrade of the specification, rather than a redesign.

Among the features to be included in Version 2 from the Systems side, the most notable are MPEG-J, MPEG-4's file format (MP4), and server interactivity.

MPEG-J refers to the use of Java within an MPEG-4 terminal. The intention for the use of Java in MPEG-4 has two basic tenets: first, the implementation of sophisticated applications, and, second, providing programmatic means through which a terminal can dynamically adapt the received content to its capabilities. The latter is particularly important when we consider that MPEG-4 does not define composition, and hence does not provide normative reference points in terms of terminal capabilities.

Figure 7 depicts the architecture through which (at the time of this writing) Java is being integrated within MPEG-4. At the bottom of the figure there is the usual system decoder model, where multiple decoders are interfaced to the delivery system via the DAI and decoder buffers, and to the compositor via composition buffers. The Java virtual machine runs on the terminal, and processes MPEG-4 applets (called MPEG-lets). These applets have access to a run-time environment, comprised of a set of Java class libraries. This set includes some

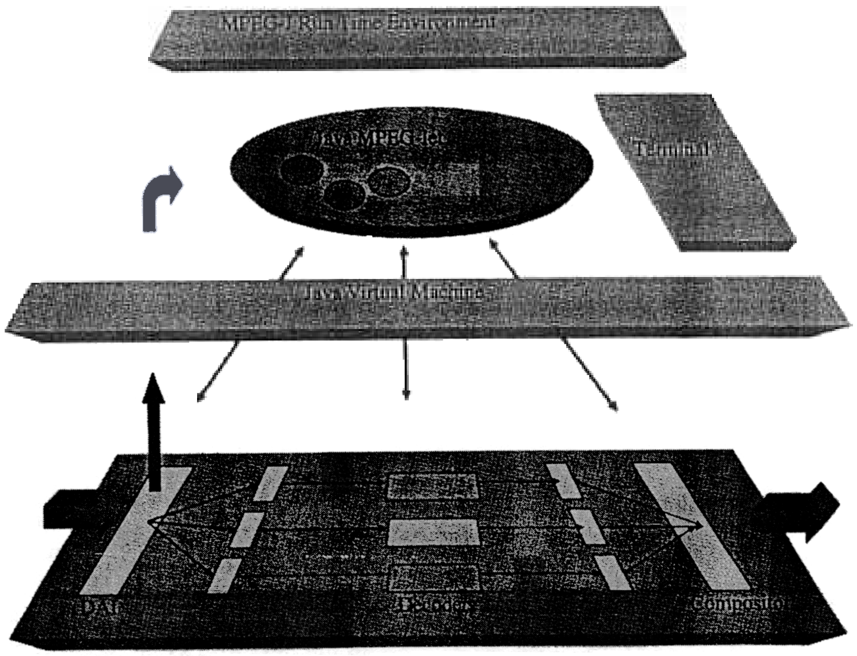


Figure 7. Use of Java within MPEG-4 (MPEG-J).

of the standard Java classes as well as MPEG-specific libraries.

The MPEG Java libraries provide functionalities for:

- application management;
- scene graph management;
- resource management;
- media decoder control;
- networking and device management; and
- terminal capability management.

It is important to note that the MPEG-J architecture explicitly avoids the use of Java in the media data path. As we see in Fig. 7, the Java code has only supervisory role. This ensures that time-critical and/or computationally intensive operations are not hindered by the interpreted and non-real-time (due to garbage collection) nature of Java.

In addition to Java support, another major addition for Version 2 is the definition of a file format. In the past, MPEG relied on the format of its own designs for multiplexed data (program and transport streams), which directly served as storage formats. Due to the significantly expanded flexibility afforded by MPEG-4, there was a clear need for a more sophisticated content interchange format that would cater to the needs of the various components of the industry (content creators, distributors, service providers, server operators, even end-users).

As a result, work on the so-called MP4 file format was started, using Apple's QuickTime specification [15] as the starting point. The objective is to define an MPEG-4 specific format, but in a way that is compatible with existing QuickTime software. The selection of QuickTime was based, to a large extent, to its capability to provide streamed versions of data files for various protocols without duplication of data (via so-called hint tracks). This implies that a server can easily stream MPEG-4 content from an MP4 file by using a hint track for the transport protocol it is using. The media data is *referenced* from the hint track, rather than being copied.

Finally, a third major addition in Version 2 is normative support for server-based interactivity. Version 1 did not define a back channel, and as a result it did not provide any direct means of sending messages or commands back to a content server. Version 2 introduces a 'ServerCommand' node, which allows the server to be a full and equal participant in the interaction model of the client terminal. The use of a node allows the routing of events to it, which then triggers the transmission of an application-defined message back to the server (with appropriate sync layer packetization). The server can then respond with a scene update, or with

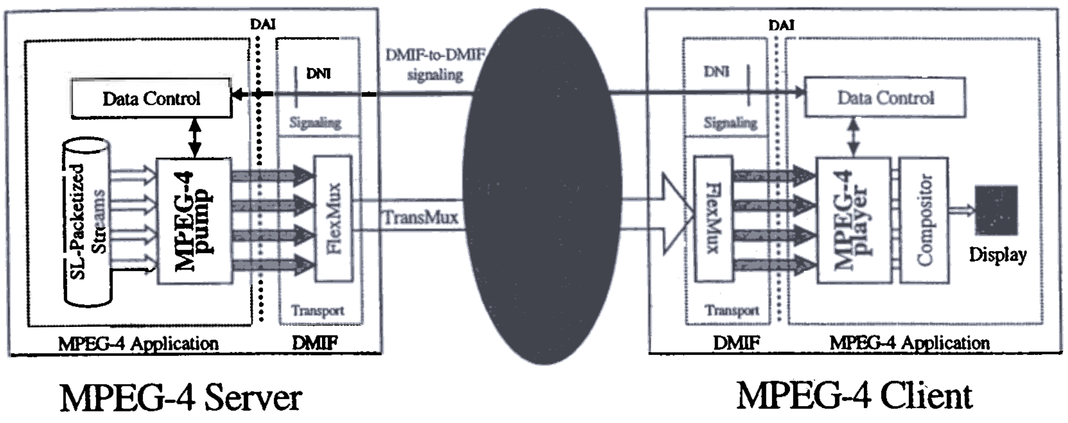


Figure 8. MPEG-4 client-server software.

whatever mechanism provided by the application designer. Note that such functionality can also be partially implemented using URLs. However, URLs are only triggered when content is being accessed, whereas server commands can be triggered by any scene event.

8. MPEG-4 Software

In order to verify its specifications and promote their acceptance in the marketplace, starting from MPEG-4 the MPEG group adopted a policy of publishing source code that implements each specification. This 'reference software' is also extremely useful for implementors who are now able to test their designs (and their interpretation of the specification) against an existing tool.

This policy has resulted in the so-called IM1 software, a collaborative development activity that provides a complete implementation of MPEG-4 Systems. (The name indicates implementation 1, as in the past there was a second implementation based exclusively on Java.) Since composition is outside the scope of MPEG-4, other parties have contributed compositors as well as hosting applications for various platforms (Windows and UNIX). In addition to a player, several other tools are also available for scene encoding, descriptor encoding, as well as multiplexing and MP4 file creation. More details can be found on the MPEG web site [5].

In parallel to the main IM1 activity, we have developed a complete MPEG-4 client-server system to investigate issues of object-based representation, real-time composition, and object multiplex scheduling.

The system consists of an MPEG-4 client and a server. The client is based on the IM1 reference software, with DMIF code provided by Xbind Inc. and 2-D composition code provided by CSELT. The client supports JPEG images, as well as H.261 video and H.723.1 audio. The server, multiplexer, and original content were developed at Columbia. The server is designed to deliver objects to the client upon demand and fully supports server itneractivity. Communication between the client and the server is performed using UDP/IP. Figure 8 shows the architecture of the system.

The complete system was demonstrated in the October 1998 MPEG meeting in Atlantic City, New Jersey, with help from Lockeed Martin Telecommunications. The server was located in Sunnyvale, California, in Lockheed Martin's laboratories, and was connected to the client via a satellite connection. At the core of the demonstration is the delivery of audio-visual objects over an IP network and decoding and composing them at the receiver The application developed provided for content selection in an interactive TV environment.

9. Concluding Remarks

MPEG-4 addresses technological issues that are extremely relevant today. There is considerable commercial interest in interactive TV and multimedia applications in general that has provided a number of alternatives to MPEG-4, in terms of architecture and functionality. In general, none of the existing solutions provides a complete toolset with features comparable to MPEG-4 under a single design.

A number of alternatives to MPEG-4 BIFS base their syntax architecture on an XML [16] syntax, while MPEG-4 bases its syntax architecture on VRML using a binary format. The main difference between the two is that XML is a general-purpose text-based description language for tagged data, while VRML with Flavor provide a binary format for a (heavily visual) scene description language.

An advantage of an XML-based approach is ease of authoring; documents can be easily generated using a text editor. However, for delivery over a limited bandwidth medium, a compressed representation of multimedia information is without a doubt the best approach from a bandwidth efficiency point of view. A textual representation of the content may still be useful at the authoring stage. Such (simplified) textual representations for MPEG-4 content, based on XML, are under consideration, while text-based tools for compiling BIFS from VRML-like text files are already available. An important consideration is that VRML has been designed for scenes that have a strong visual component, and thus provides very powerful tools for scene construction.

At the time the MPEG-4 standard was published, several specifications were providing semantics with an XML-compliant syntax for multimedia representation in specific domains. For example, the W3C HTML-NG was redesigning HTML to be XML compliant [17], the W3C SMIL working group has produced a specification for 2D multimedia scene descriptions [18], the ATSC/DASE BHTML specifications were working at providing broadcast extensions to HTML-NG, the W3D X3D requirements were investigating the use of XML for 3D scene description, whereas W3C SVG was standardizing scalable vector graphics also in an XML compliant way [19].

MPEG-4 is built on a true 3D scene description, including the event model, as provided by VRML. None of the XML-based specifications currently available reaches the sophistication of MPEG-4 in terms of composition capabilities and interactivity features. Furthermore, incorporation of the temporal component in terms of streamed scene descriptions is a non-trivial matter. MPEG-4 has successfully addressed this issue, as well as the overall timing and synchronization issues, whereas alternative approaches are lacking in this respect.

Finally, the industry is already providing compelling solutions for the representation of multimedia content. Indeed, functionalities like those offered by MPEG-4 have already appeared in products such as games and CD-ROM authoring applications, long before the standard was even finalized. There is always a time lag between the functionality of the more advanced products in a domain and relevant open technology standards. However, as soon as a standard appears that solves the needs of the industries involved, a range of diverse interoperable product offerings and their market deployment is made possible. Each player may then invest in its core business, seeking to gain competitive advantage in areas not covered by the standard. It is therefore very unlikely that a proprietary format for multimedia representation can compete successfully, in the long run, with an open standard like MPEG-4 when universal access is required at both national and international levels.

A final consideration is that MPEG-4 has been designed from the ground-up to address interactive audiovisual applications. Most other competing solutions originate from a Web-centric architecture. As a result, while they may provide solutions that are appropriate for the Web, it is not necessarily true that they cater to the needs of the video and audio content industry. In the case of interactive TV, for example, several companies are currently making efforts to integrate Web access to set-top boxes. While this is an interesting and useful application, it is not the same as providing interactive TV content. While the two can share the same screen (similarly to a word processing and a spreadsheet program running on the same computer), they are not the same application. This is a point that is often missed when considering architectures that combine digital video and audio and interactivity.

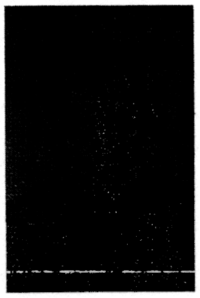
Acknowledgments

The development of an international standard is a group effort involving a large number of individuals from all around the world. The author would like to particularly acknowledge the critical contributions of the following individuals in the design of MPEG-4 Systems: Olivier Avaro (France Telecom), Carsten Herpel (Thomson Multimedia), Julien Signes (France Telecom), Liam Ward (Teltec), and Ganesh Rajan (General Instrument).

References

1. ISO/IEC 14496-1, Coding Of Audio-Visual Objects: Systems, Final Draft International Standard, JTC1/SC29/WG11 N2501, Oct. 1998.
2. ISO/IEC JTC1/SC29/WG11 N2611, MPEG-4 Systems Version 2 WD 5.0, Dec. 1998.

3. A. Puri and A. Eleftheriadis, "MPEG-4: An Object-Based Standard for Multimedia Coding," in *Visual Information Representation, Communication, and Image Processing*, C.-W. Chen and Y.-Q. Zhang (Eds.), New York: Marcel Dekker, 1999.
4. ISO Web Site, `http://www.iso.ch`.
5. MPEG Web Site, `http://www.cselt.it/mpeg`.
6. B.G. Haskell, A. Puri, and A.N. Netravali, *Digital Video: An Introduction to MPEG-2*, London: Chapman and Hall, 1997.
7. O. Avaro, A. Eleftheriadis, C. Herpel, G. Rajan, and L. Ward, "MPEG-4 Systems Overview," *Multimedia Systems, Standard, Networks*, A. Puri and T. Chen (Eds.), New York: Marcel Dekker, 2000.
8. A. Eleftheriadis, "Flavor: A Language for Media Representation," in *Proceedings, ACM Multimedia '97 Conference*, Seattle, WA, Nov. 1997.
9. Flavor Web Site, http://www.ee.columbia.edu/flavor.
10. C. Herpel, A. Eleftheriadis, and G. Francescini, "MPEG-4 Systems: Elementary Stream Management and Delivery," *Multimedia Systems, Standard, Networks*, A. Puri and T. Chen (Eds.), New York: Marcel Dekker, 2000.
11. J. Signes, Y. Fisher, and A. Eleftheriadis, "MPEG-4: Scene Representation and Interactivity," *Multimedia Systems, Standard, Networks*, A. Puri and T. Chen (Eds.), New York: Marcel Dekker, 2000.
12. R. Koenen, "MPEG-4: Multimedia for our time," *IEEE Spectrum*, vol. 36, no. 2, 1999, pp. 26–33.
13. C. Herpel and A. Eleftheriadis, "MPEG-4 Systems: Elementary Stream Management," *Signal Processing: Image Communication, Tutorial Issue on the MPEG-4 Standard*, vol. 15, nos. 4–5, 2000, pp. 299–320.
14. VRML Web Site, `http://www.vrml.org`.
15. QuickTime Web Site, `http://www.apple.com/quicktime`.
16. Extensible Markup Language (XML) 1.0, T. Bray, J. Paoli, and C.M. Sperberg-McQueen (Eds.), February 10, 1998, `http://www.w3.org/TR/REC-xml`.
17. XHTML™ 1.0: The Extensible HyperText Markup Language. A Reformulation of HTML 4.0 in XML 1.0. W3C Working Draft, February 24, 1999, `http://www.w3.org/TR/WD-html-in-xml/`.
18. Synchronized Multimedia Integration Language. (SMIL) 1.0 Specification, June 15, 1998, `http://www.w3.org/TR/REC-smil`.
19. Scalable Vector Graphics (SVG) Specification. W3C Working Draft, February 11, 1999, `http://www.w3.org/TR/WD-SVG`.



Alexandros Eleftheriadis was born in Athens, Greece, in 1967. He received the Diploma in Electrical Engineering and Computer Science from the National Technical University of Athens, Greece, in 1990, and the M.S., M.Phil., and Ph.D. degrees in Electrical Engineering from Columbia University, New York, in 1992, 1994, and 1995 respectively. Since 1995 he has been in the faculty of the Department of Electrical Engineering at Columbia University (currently as an Associate Professor), where he is leading a research team working on media representation, with emphasis on multimedia software, video signal processing and compression, video communication systems (including video-on-demand and Internet video), and the mathematical fundamentals of compression. He is also co-principal investigator of the ADVENT Project (http://www.ee.columbia.edu/advent), an industrial affiliates program at Columbia University that is performing research on all aspects of digital video representation, communication, and description. Dr. Eleftheriadis is a member of the ANSI NCITS L3.1 Committee and the ISO-IEC JTC1/SC29/WG11 (MPEG) Group, that develop national and international standards for audio and video coding. He also served as the Editor of the MPEG-4 Systems (ISO/IEC 14496-1) specification. He has authored more than 60 publications in international journals and conferences and holds 6 patents (11 pending). His awards include an NSF CAREER Award. He is in the editorial board of the Multimedia Tools and Applications Journal, has served as a guest editor, committee member, and organizer for several international journals and conferences, and Dr. Eleftheriadis is a member of the IEEE, the ACM, the AAAS, and the Technical Chamber of Greece.
eleft@ee.columbia.edu