Adding Delivery support to MPEG-Pro, an Authoring System for MPEG-4

Frederic Bouilhaguet, Cyril Concolato, Souhila Boughoufalah, Jean-Claude Dufourd,
ENST Paris Dept ComElec, 46, rue Barrault, 75013 Paris

E-mail: { bouilhaguet, concolat, souhila, dufourd }@ enst.fr

Abstract

MPEG-Pro, an MPEG-4 authoring software prototype has been designed to integrate all main authoring capabilities from audio/visual encoding to the final delivery of a complete MPEG-4 application. The storage format, named MP4, has been defined for MPEG-4, but adapted techniques for specific delivery scenarios that are compliant with this format are still in course of proposition and definition. We introduce a solution to author and store MPEG-4 content with a better data accessibility for the streaming over RTP/IP scenario. By respecting scrupulously the MP4 constraints, this content remain reusable by any MPEG-4 compliant tool. We describe how its implementation is integrated in MPEG-Pro whose modular architecture allows a large scope of extensibility.

1 Introduction

MPEG-Pro is a software dedicated to authoring 2D multimedia content compliant with the latest MPEG-4 Systems specifications. Its architecture is voluntarily open. A set of components has been integrated since its first version. Some of them implement tools of MPEG-4 Systems, but some others add enhancement techniques to face some limitations of the MPEG-4 standard in the context of multimedia authoring [1]. The MP4 file format is specified to store MPEG-4 content that enables at the same time its edition for authoring, its preparation for streaming over networks and its local playability. It has strict rules to ensure the interchangeability of multimedia contents. But it also offers a high level of flexibility to optimize storage dedicated to different scenarios of either interactive functionality or delivery. In this paper, we present the latest evolution of MPEG-Pro taking into account this flexibility. This evolution consists in giving the user the possibility to interleave the elementary streams of the MP4 file in different ways. We expose our own algorithm for interleaving content and its implementation. Then, we show some advantages of interleaving MPEG-4 content for several delivery scenarios and how our method is concerned with the specific multiplexer tool of MPEG-4 Systems, named the FlexMux.

2 Overview of MPEG-4 Systems

2.1 The 3 layers

The MPEG-4 architecture is distributed along the definition of three adjacent layers : the Compression Layer, the Synchronization Layer, and the Delivery Layer [3]. In the Compresssion Layer, elementary streams (ES) are the basic abstraction for any streaming data source. An ES is organized in access units (AU), the smallest elements that can be attributed to individual time stamps. Coding techniques of media ESs (audio and video) have been improved by MPEG-4 Audio and Visual groups, but the main enhancement in MPEG-4 compared to its predecessors MPEG-1 and 2 is the definition of control ESs by the MPEG-4 Systems group. These control ESs are the object descriptors (OD) stream and the scene description (BIFS) stream. An MPEG-4 terminal plays an object-based MPEG-4 presentations by receiving, decoding and rendering all ESs. From its encoder to its decoder, each ES is conveyed in the Sync Layer as Sync layer-Packetized Streams (SPS). This packetized representation provides timing and synchronization information. The term Delivery Layer is used as a generic abstraction of any existing transport protocol stack that may be used to transmit MPEG-4 content. As a wide variety of delivery mechanisms exist inside this layer, MPEG-4 Systems defines just a unique interface (DAI = DMIF Application Interface) to access this layer.

2.2 OD framework and scene description

The location and properties of ESs in a MPEG-4 presentation are described by a dynamic set of object descriptors (ODs). Similar to any audio-visual content, ODs are transported in a dedicated ES, called the OD ES. Within the OD ES, it is possible through object descriptor commands (OD commands) to dynamically convey, update and remove complete ODs. Updates are time stamped to indicate the instant in time they take effect. The binary coding of the OD stream is specified by MPEG-4 Systems.

The session description provided by the ODs is accompanied by a dynamic scene description, again conveyed through one or more ESs. This scene description, initially based on VRML, uses a parametric approach (BIFS - Binary Format for Scenes). Content is identified in terms of audio-visual objects. The spatial and temporal location of

each object is defined by BIFS. The description consists of an encoded tree of nodes with attributes and other information (including event sources and targets). The scene description can evolve over time by using scene description updates (BIFSUpdates commands). The BIFSUpdates commands are time stamped and contained in BIFS AUs. Interactivity mechanisms are integrated in BIFS. They are defined by linked events from sources to targets nodes, called routes, and special nodes, called sensors, that can trigger events on specific conditions.

2.3 FlexMux : the tool for multiplexing

For applications where the desired transport facility does not fully address the needs of a service according to the number of transport streams, a multiplexing technique, called FlexMultiplexing, has been defined. The FlexMux tool is a flexible multiplexer that accommodates interleaving of SL-packetized streams. The FlexMux tool provides identification of SL packets originating from different elementary streams by means of FlexMux Channel numbers. Each SL-packetized stream is mapped into one FlexMux Channel. FlexMux packets with data from different SL-packetized streams can therefore be arbitrarily interleaved. The sequence of FlexMux packets that are interleaved into one stream are called a FlexMux Stream.

2.4 MP4 : the file format

The MP4 file format, initially based on *QuickTime* [4], is designed to contain the media information of an MPEG-4 presentation in a flexible, extensible format that facilitates interchange, management, editing, and presentation of the media. The media-data in MP4 is not encapsulated in frames with description headers. The meta-data is used to describe the media data characteristics (media type, times stamps, size ...) by reference, not by inclusion. The meta-data is stored in structure, called 'moov', while media-data is stored by chunks in a container structure called 'mdat'. The 'moov' meta-data structure is subdivided in substructures named atoms. The offset position of an AU in MP4 is obtained by a computing mechanism using the 'stsc' (*Sample to Chunk*), 'stco' (*Chunk Offset*), and 'stsz' (Sample Size) atoms.

3 Architecture of MPEG-Pro

MPEG-Pro is an authoring software to help authors creating MPEG-4 contents from the end-user interface specification phase to the cross-platform MP4 file. It is based on timeline editing. All content creation principles used in this application match closely MPEG-4 Systems. Two main module compose the architecture of MPEG-Pro as illustrated by the *figure 1*. They are called Core and Extensions The Core is composed of components that communicate together and with the Extensions. An Extension is a component that can communicate only with Core components.

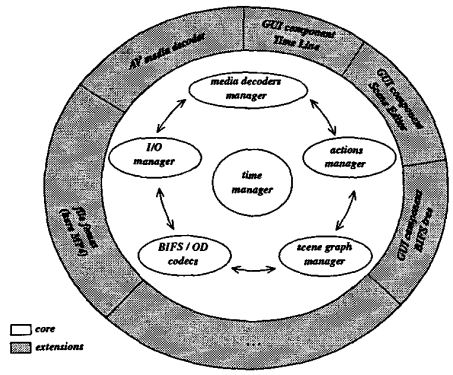

Figure 1 : the architecture of MPEG-Pro

The Core contains functions to author the OD ES and the BIFS ES. It consists in encoding and decoding *BIFS* in the OD/BIFS codecs component and maintaining an internal form for BIFS and OD for the scene and updates, exposing a scene modification API (scene graph manager) to the rest of the application. A scene controller maintains its BIFSUpdates list and get the ODs of its media nodes from the OD controller. The OD controller maintains a list of the ODUpdates commands. Each author modification is registered as one or several update commands, and inserted after consistency checking in the BIFS and OD controllers.

An extension called File Format is dedicated to storage and retrieval in the MP4 format. It communicates with the rest of the application through the I/O Manager module of the Core. It reads MP4 files into memory, provides an API to access the meta-information and the ESs, imports raw streams by creating the needed meta-information, and writes the edited MPEG-4 presentation into MP4 files. When reading MP4 content, it provides stored AUs to the *media* (audio/video) and *control* (*BIFS/OD*) decoders.

4 A method to interleave content in MP4

4.1 Interleaving principles in MP4

One of the issues to play back an MP4 file in real time is the number of file seeks that must be performed. It is possible to arrange the data in an *MP4* file to minimize any seeks during the course of normal playback. This is accomplished by interleaving the ES data. The AUs of an ES are stored in the 'mdat' container *MP4* by chunks, while its description is stored in the 'moov' meta-data structure. When the ESs are separate, all AUs of each ES are stored in only one chunk. Interleaving ESs consists in splitting the chunks and

mixing them in the 'mdat' container, and reporting the modifications in the 'moov' structure. No method for interleaving is specified by MPEG-4 Systems. Any proposed method must respect the constraints of the MP4 structure.

4.2 Our method

a) Grouping the ESs

Since an MPEG-4 presentation may involve a large number of objects, interleaving all the corresponding ESs may require strategies depending on the interactive scene scenario. So we decided to enable the author to group ESs with similar interleaving requirements. From a Streams List window, the author can create ES groups and add any ES to them by simply dragging and dropping its icon ,as illustrated in *figure 2*. By default, a newly created ES is first included in the main ES group related to the Streams List. For each ES group, a specific parameter quantify the degree of interleaving. This parameter, called *MAX_chunkDuration,* fixes a duration limit of the MP4 chunks for the MP4 Tracks which are associated to the ESs. We define the duration of an MP4 chunk as the difference between the DTS of the last and first AU inside the chunk :

$$chunkDuration = DTS(lastAU) - DTS(firstAU)$$

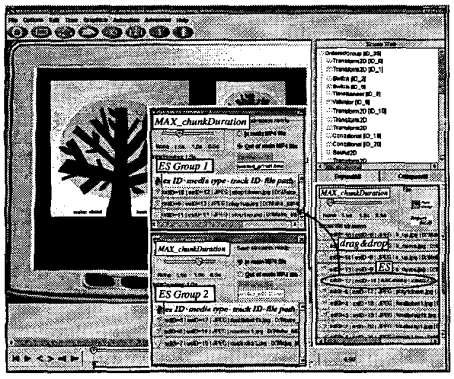

Figure 2 : user interface in MPEG-Pro for grouping ESs

b) Interleaving the ESs by groups

The AUs are interleaved by circularly scanning all ESs that belong to the same group. The ESs are scanned in the order of their appearance in the ES group window. The user can change this order by moving the ESs icons in the list. A weighted round-robin scheme, as illustrated on the *figure 3,* has been implemented to achieve the selection of the AUs from the different ESs. At each cycle, the weight associated to an ES is the number of consecutive AUs to be selected and written in an 'mdat' atom of the MP4 output file. A temporal window whose width is set by the interleaving parameter *MAX_chunkDuration* allows to determine this number. If *cycleNum* is the number of the cycle, an AU is selected if it fulfils the following conditions :

$$(cycleNum-1)\times MAX\_chunkDuration \leq DTS(AU)$$

and

$$DTS(AU) < cycleNum \times MAX\_chunkDuration$$

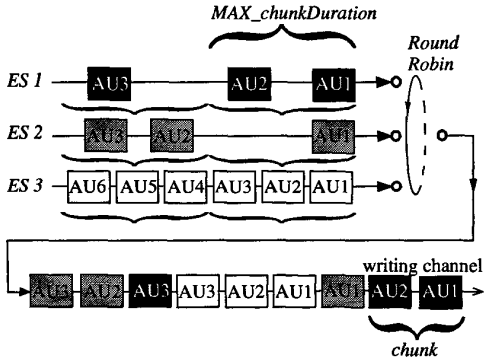

Figure 3 : interleaving scheme for an ES group

c) Storing in MP4

Having grouped and interleaved the ESs, the associated 'moov' structure must be calculated. As MP4 gives the possibility to store the data in external files and reference them in the meta-data, the author can choose in MPEG-Pro to store any ES group in a specific file, or he can keep it in the same file as the meta-data. The storing procedure ends by writing separate ES groups in destination files, as illustrated in *figure 4*, and synchronising the meta-data atoms concerned by the access units positions.

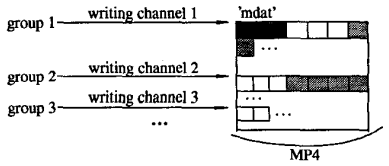

Figure 4 : writing all the groups in a MP4 structure

5 Benefits of interleaving in MP4

5.1 Local scenario

Viewing an MPEG-4 presentation in a local context (i.e. directly from the file and not over streamed SL packets) is an important application. It will be used as soon as MPEG-4 contents will be available on CD or DVD ROM. On such a storage medium where files are never fragmented, interleaving is important because seeking over hundreds of megabytes, as illustrated on *figure 5*, may slow the reading process and cause player freeze.

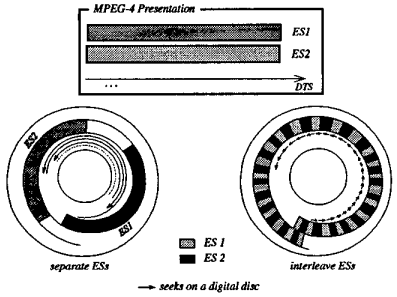

Figure 5 : effects on the seeks

5.2 HTTP "streaming"

Given the number of HTTP server active over the Internet, it is interesting to try to make use of these to achieve simple streaming without any QoS requirement. To be streamable by HTTP, the file must be constructed in playing order so that the client does not need to download the whole file before starting to play it. When interleaving data, MPEG-Pro puts the MP4 meta-data ('moov' atom) at the top of the file and order the AUs in increasing DTS order. If there are many ES groups, each group is placed in a separate file. If bandwith is sufficient, the terminal may start playing as soon as the a few seconds of buffer are ready for each ES group. The terminal needs to be HTTP aware and to wait for rebuffering if one of the ES groups is "late".

5.3 Flexmuxed stream over RTP

Current multimedia streaming solutions that use RTP [5] [6] [7] do not require a lot of RTP sessions because presentations are made of two stream (audio and video). MPEG-4 is about to change this situation. An MPEG-4 presentation can possibly have a lot of ESs: one per object of the presentation, plus one for the BIFS and one for the ODs. If the same streaming method is kept (one media stream for one RTP session), this leads to a high-level of complexity when handling all the corresponding RTP sessions. Therefore, multiplexing the ESs into one RTP stream using the FlexMux tool has been proposed by MPEG-4. Two different modes of operation of the FlexMux providing different features and complexity are defined : *Simple* mode and *MuxCode* [3; section 12]. In the MuxCode mode one or more SL packets are encapsulated in one FlexMux packet. The configuration information that defines the FlexMux payload is stored in the MuxCode Table. If the AUs within an ES have constant length and if the MuxCode Table is built in such a way that it matches the way the AUs are interleaved in the MP4 file, then building RTP packets to stream all the ESs using the FlexMux is really easy. The advantage is that, since the AUs of an RTP packet are neighbouring each others, the construction of this latter consists in copying a block of contiguous data according to the MuxCode entries.

6 Conclusion

In this document, we presented how a new interleaving tool has been added to MPEG-Pro, our MPEG-4 authoring tool in progress. We explained the advantages of such a tool in three delivery scenarios: local playback, HTTP "streaming" and streaming of Flexmuxed content over RTP/IP, to avoid the overhead of having many RTP streams. Furthermore exploiting the MPEG-4 architecture, a next step to this work would be the implementation of the hinting process to store the Flexmux information. Implementation of the dedicated hinter is pending on the definition of the RTP payload format for Flexmuxed streams which is in progress between the IETF and MPEG.

Bibliography

[1] Boughoufalah, Dufourd, Bouilhaguet (ENST Paris) "MPEG-Pro, an Authorong System for MPEG-4 with Temporal Constraints and Template Guided Editing", Proceedings of the ICME 2000 International Conference, New York, Aug 2000.

[2] Bouilhaguet, Dufourd, Boughoufalah (ENST Paris) "Interactive Broadcast Digital Television- The OpenTV Platform versus the MPEG-4 Standard Framework », Proceedings of the ISCAS 2000 International Symposium, Geneva, May 2000

[3] ISO/IEC 14496-1:2000 MPEG-4 Systems October 200

[4] Apple Computer Inc. "QuickTime File Format - Publication Draft », June 2000

[5] Microsoft, online www.microsoft.com/windows/windowsmedia

[6] Apple Computer, online www.publicsource.apple.com/projects/streaming

[7] Realnetworks, online www.real.com

[8] Schulzrinne, Casner, Frederick, Jacobson RTP: A Transport Protocol for Real Time Applications RFC 1889, Internet Engineering Task Force, January 1996

[9] Kalva, Huard, Tselikis, Zamora, Cheok, Eleftheriadis "Implementing Multiplexing, Streaming and Server Interaction for MPEG-4", Proceedings of IEEE Transctions on Circuits and Systems for Video Technology, Vol 9, No 8, December 1999