INTERNATIONAL ORGANIZATION FOR STANDARDIZATION
ORGANISATION INTERNATIONALE DE NORMALISATION
ISO/IEC JTC1/SC29/WG11
CODING OF MOVING PICTURES AND ASSOCIATED AUDIO

ISO/IEC JTC1/SC29/WG11
MPEG97/M2062
Bristol, UK

Title: **Stored File Format for MPEG-4**
Source: **Alexandros Eleftheriadis and Hari Kalva, Columbia University, USA**
Atul Puri and Robert Schmidt, AT&T Research, USA
Group: **MPEG-4 Systems**
Date: **April 1, 1997**

1. Introduction

The MPEG-1 and MPEG-2 standards specify bitstream representations for Audio, Video and Systems. Although they are intended to support interactivity, the flexibilities they offer are rather limited. Among the reasons for these limitations is the lack of a stored file format defined by MPEG. A number of ad hoc and incompatible extensions for file formats are currently in use such as .mpg, m1v, m2v, .m2s etc. Since MPEG-4 offers opportunities for much more flexibility in terms of types of audiovisual objects, editing of objects, composition of objects and user controlled presentation, it seems even more necessary to standardize a flexible and extensible file format. In other words, defining a standardized MPEG-4 file format will facilitate the following:

- Random Access to AV Objects
- Editing of AV Objects - insertion, deletion, modification
- Random Access to Composition Data
- Editing of Composition Data
- Unified representation of MPEG-1, MPEG-2: audio, video and systems data files, and coded MPEG-4 data

2. MPEG-4 File Format

MPEG-4 files can use .mp4 as the file identifying extension. All stored AV Objects related to a session and conforming to MPEG-4 coding reside in such a file. Since the file format needs to allow efficient editing of the session, and yet should be conservative with bandwidth for streaming access over the network, a mechanism is provided in the file header to exclude editing information. We now discuss the details of a proposed format for .mp4 files; this format is shown in Figure 1.

A stored stream (file) consists of a stream header containing global information about the stream, a header checksum, and some number of segments containing the AV Object data. The AV Objects are interspersed within the stream segments, and each segment contains a header describing the AV Objects located within the header, except in the case where the file is intended for streaming only.

Stream Header

The first byte of the Stream Header contains the file type definition. This byte describes the contents of the file according to the following table.

Bit 0 – If set indicates an streaming file (editing information removed)
Bit 1 – If set indicates a BIFS file location is present
Bit 2 – If set indicates that there are Physical AV Objects present in the stream
Bit 3 – If set indicates that there are Logical AV Objects present in the stream
Bit 4 – Undefined
Bit 5 – Undefined
Bit 6 – Undefined
Bit 7 – Undefined (*Note: can be used to signal extensions*)

Following the file type field is a 1 byte code describing the profile/level of the entire stream. This will allow a decoder to determine if it is capable of handling the data in the stream. After the stream profile field is the optional BIFS location. If this is present it is represented by a 1 byte string size field followed by the string defining the file location of the BIFS file (without a terminating null character '\0'). The next portion is the optional Physical Stream Object Table which catalogs a description of all objects in the stream that are physically present in the file. The stream header next contains an optional Logical Object Table, which catalogs the location of all stream objects not physically present in the file. Next in the stream header is a 2 byte segment count field, and a checksum field for checking the integrity of the received header information.

Physical Object Table
The Physical Object Table is optional, and when present it is contained in the Stream Header. It consists of a 2 byte AV Object count, indicating the number of AV Objects in the stream, followed by a 2 byte ID and 1 byte profile/level descriptions for each AV Object present in the Stream. If the file is an editable file, each AV Object description will contain an additional 2 bytes to indicate the segment in which the AV Object first occurs in the stream.

Logical Object Table
The Logical Object Table is also optional, and when present it is contained in the Stream Header. It consists of a 2 byte AV Object count indicating the AV Objects that are part of the session, but not physically present in the file. This is followed by a 2 byte AV Object ID, a 1 byte object location string length, and the string indicating the location (file or URL) of each AV Object in the table.

Stream Segment
If the stream is large, it is divided into several segments, with the number of segments indicated in the stream header There are two types of stream segments, and each is uniquely identified by a 32 bit start code. The first type of segment contains AV Object data. It is indicated by the start code 0x000001B9. The second type of segment contains composition information. It is indicated by the start code 0x000001BF. Following the segment start code is a 2 byte segment size field. If the file is an editable file, and the segment contains AV Object data, the size field is followed by a segment AV Object table containing a 2 byte object count field, which indicates how many objects are in the segment, and a 6 byte field for each object in the segment. The first 2 bytes are the AV Object ID, the next 2 bytes indicate the offset to the starting point of that AV Object in the segment, and the last 2 bytes are the size of the object. The two most significant bits of the size have the following meaning:

00 - complete object
01 - 1st segment of a split object next segment follows; look in the object tables
10 - Last segment of a split object;

11 - intermediate segment of a split object; look in the object table to locate the next object segment.

The 14 least significant bits gives the object segment size.

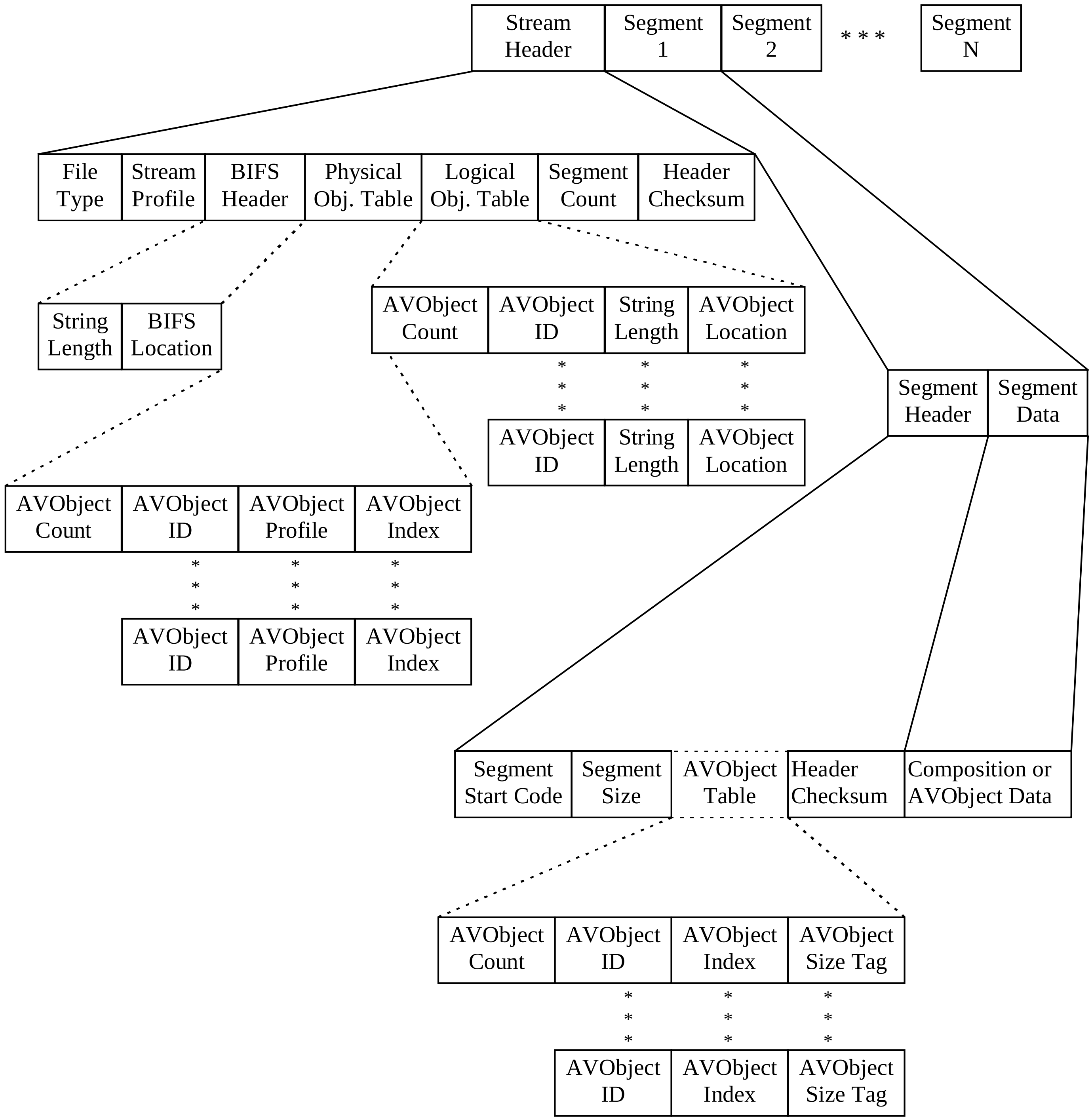

Fig 1. Bitstream structure

All object segments are listed in the object table. This also allows more than one object (instance) with the same ID to be present in the same stream segment. The segment header is followed by a header checksum. The data portion of the segment will contain AV Object Data for a AV Object type segment, or composition data such as scene graphs, node updates, and stream-node mapping tables, if the segment is a composition type.

Number of segments and segment size: 2 bytes for segment size and 2 bytes for no. of segments in the stream are allowed. This gives a max stream size of 4 Giga bytes. This should be sufficient for most applications. However to support larger files (e.g., in a Studio Profile??) an extensible code can be used. This convention will not add any overhead in case of smaller files and also supports large file sizes if necessary.

> **Extensible code:** An extensible code is used to accommodate more than 64 K segments or a segment size larger than 64 K. The following convention shall be used in such cases. A value of 0XFFFE corresponds to: 64K - 1 + the value of the two bytes following immediately. This can be extended following the same convention.

As shown in Figure 2, an adapter can be employed to strip the editing information from the file and indicate it as file intended for streaming.

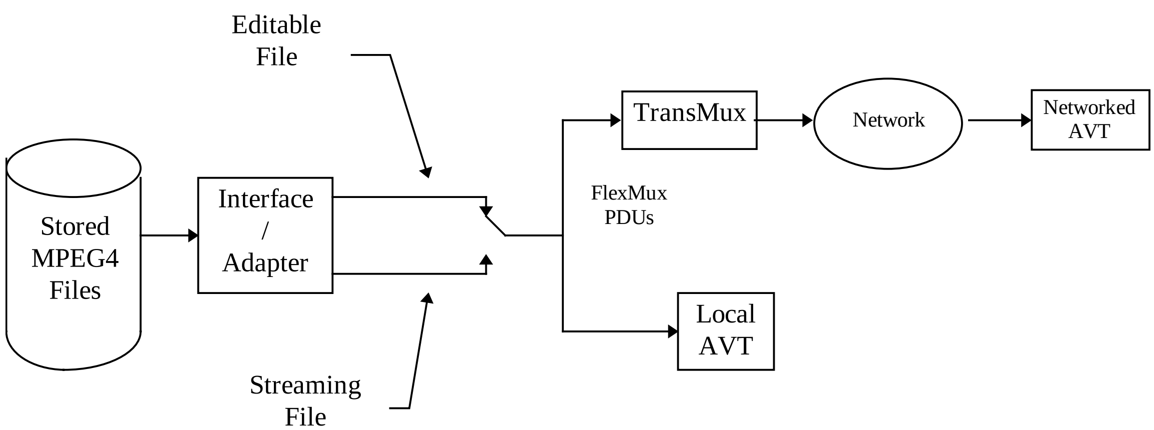

Fig 2. Serving Stored objects to Audio Visual Terminals (AVT)

3. Discussion

Although the proposal still needs significant refinement, it introduces the benefits, basic concepts and a proposed draft of the file format for MPEG-4. It is primarily intended to inspire a discussion on the topic; we would like to take any feedback and suggestions into account to improve on the proposed description.