A STANDARD TARGET DECODER MODEL FOR MPEG-4 FLEXMUX STREAMS

Arun Ramaswamy

Vela LP
5733 Myerlake Circle
Clearwater, FL 33760, USA

ABSTRACT

MPEG-4, like other MPEG standards is generic and universal in the sense that it specifies compressed bitstream syntax. This, in effect, unambiguously defines the decompression process and the decoder architecture. Further, to ensure interoperability between encoders and decoders, it is essential that a Standard Target Decoder model (STD) be well defined. Since STD is a hypothetical model in the encoder architecture, multiplexors can create compliant streams without having to base its design around any particular decoder. At the same time, compliant decoders should have the minimum buffers as mandated by the STD and hence should be able to decode compliant streams. This paper proposes a STD model for MPEG-4 FlexMux framework.

1. INTRODUCTION

MPEG-4, is aimed towards low bit rate applications such as video-over-Internet and mobile multimedia, interactive video games. MPEG-4 provides a more integrated and flexible multimedia solution than MPEG-1 and MPEG-2. For example, for the content creators, MPEG-4 allows the integration of natural video with 2-D and 3-D graphics with animations, text, natural and synthetic audio. For network service providers, MPEG-4 provides a generic quality of service (QoS) parameter set applicable for the diverse MPEG-4 media For the end user, the standard provides a new level of user interactivity.

To facilitate these new features, MPEG-4 has adopted an "object-based" coding scheme, where each audio/video (A/V) object is isolated from a scene and coded potentially at different bitrates. These streams may be multiplexed together or transmitted individually. Also accompanying these A/V streams, is a "Scene Description" stream that allows the final scene composition using the various decoded A/V objects. At the receiver, the objects are demultiplexed, decoded and presented with precise timing to produce a rich multimedia presentation for the end user. For synthetic multimedia objects, (unlike natural video), a process of composition follows the decode process. The encoder conveys the timing for the decode, presentation and composition by means of timestamps, which are samples of its clock. These time stamps are embedded in the stream and are recovered by the receiver. In this classical communication paradigm, the end to end delay between the encoder and decoder is assumed to be constant. The network is assumed to produce zero jitter, Figure. 1

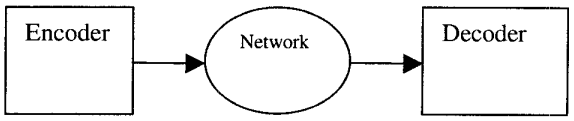


Figure 1. The Encoder/Decoder Model. The end to end delay between the encoder and decoder is assumed to be constant. The network jitter is assumed to be zero.

MPEG-4 applications can be classified into two main categories. One set of applications are "Pull" oriented, e.g. Internet based applications. Here the decoder pulls data from the source. The decoder directly determines the rate of consumption of data. The second category is more for broadcast applications, which are "Push" driven. Here, a server at a deterministic rate streams data to the decoder. The decoder operates with a fixed buffer size. The clock is recovered from the stream. This is used to synchronize the decoder's clock, thus matching the consumption rate of the decoder with the input-streaming rate.

For a Push model, decoders that are implemented in hardware often operate with a fixed buffer size. Hence efficient buffer management and a precise timing model is critical. To ensure universal interoperability, a hypothetical mathematical decoder model is needed on the encoder side to create compliant streams. This model aids the design of multiplexors, which combine audio, video and other multimedia streams in a manner, such that the streams do not cause overflow or underflows of the decoder buffers on playback. This contribution proposes a standard decoder model (STD) for such applications. The rest of the paper is arranged as follows. The encoder and the structure of the stream model are identified in Section 2. The Section 3 details the proposed STD model. Conclusions are presented in Section 4.

2. ENCODER MODEL

A STD can only be proposed with a specific stream structure in mind. MPEG-4 proposes the usage of an optional multiplex tool called FlexMux. To create these streams the following is done,

0-7803-5471-0/99/$10.00©1999 IEEE

Fig. 2. First the Elementary streams (ES) which contain the raw compressed data of multimedia objects, are packetized into Sync Layer packets. Each coded representation of an object is associated with one or more timestamps. These timestamps include the Decode timestamps (DTS), Presentation timestamps (PTS) and the Composition timestamps (CTS). The function of these timestamps is to synchronize the presentation of various objects by informing the decoder, when to decode, composite or present the object. The timestamps of the various objects are samples of the object encoder clock. In a broadcast environment, where audio/video synchronization is imperative, the Audio/Video Clocks at the Encoder are locked. Next, these packetized streams are combined or multiplexed together into the MPEG-4 specified FlexMux streams. There also exists a dedicated ES stream, which contains the samples of the Encoder clock. This stream is called the OCR (Object Clock Reference) stream. The purpose of this stream is to synchronize the decoders clock with that of the encoder. Further, a transport multiplexing (TransMux) tool could map these FlexMux streams onto some transport mechanism. Such tools may include MPEG-2 transport streams, RTP etc. The definition and recommendations for TransMux is however outside the scope of MPEG-4 and dealt adequately in other standards bodies. To aid the design of a FlexMux multiplexor, which may combine audio, video and other multimedia streams in a manner, such that the streams do not cause overflow or underflows of the decoder buffers on playback, a STD Model is needed. This model is described next.

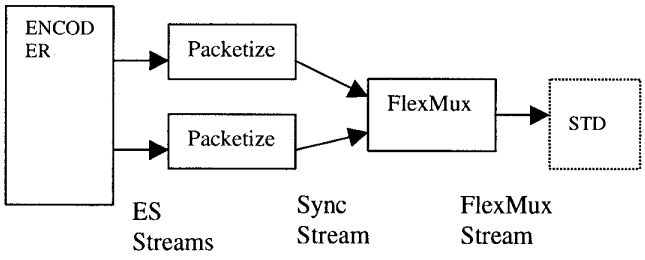
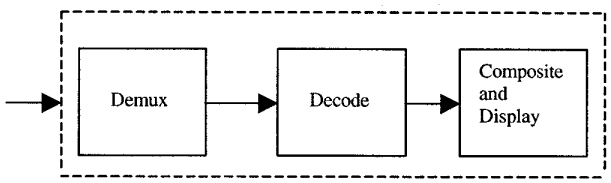


Figure 2. Multiplex Model. The Elementary streams are packetized, multiplexed, to create the MPEG-4 FlexMux Stream. In front of the Encoder is the hypothetical decoder called the Standard Target Decoder (STD)

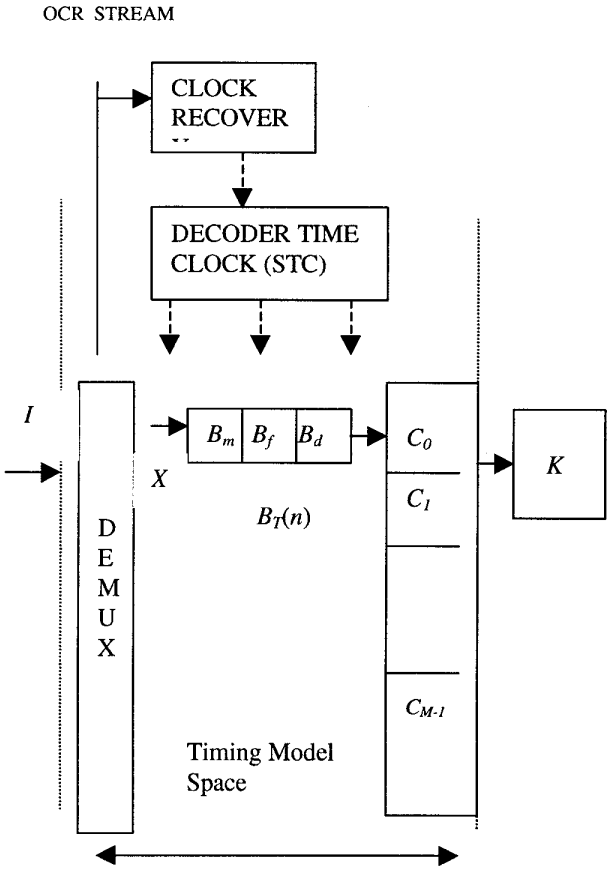
3. STD MODEL

After the encoded stream reaches the decoder via a network, overall data flow within a decoder is shown in Fig. 3.



FlexMux Stream

Figure 3. The FlexMux stream is first demultiplexed, then decoded and then displayed to the end user.



I: Input FlexMux stream
$B_T(n)$: Total decoder buffer for *n*th elementary stream
B_m: Decoder multiplex buffer
B_f: Decoder FlexMux buffer
B_d: Decoding buffer
X: *n*th Packetized Elementary stream after being demultiplexed,
C_0 to $C_{(M-1)}$: *M* Composition memory units
K: Display buffer, outside the scope of the STD

Figure 4. The STD Model. The timing model space involves the transfer of data between the demultiplexor, the decoder buffers and the composition memory units.

Referring to Fig. 3, the FlexMux stream is first demultiplexed. Next it is decoded and then finally composited and presented to the end user. A compliant encoded stream should manifest no artifacts on playback on a compliant decoder. A compliant decoder, among other features has at least the minimum amount of buffering, as modeled by the STD. A real decoder may actually have more than the minimum to compensate for the network jitter. The STD model is shown in Figure 4 and explained in the following subsections.

3.1 Inputs

A FlexMux stream, I is assumed to be input to the STD. If I contains interleaved packets of different object streams, then a demultiplexor may be employed to demux the composite FlexMux stream into the individual elementary streams, which are then directed to the various dedicated buffers. Input to each decoder buffer are packetized elementary stream packets belonging to an individual elementary stream (single object), X. The demultiplexor also separates the OCR stream for the object being decoded, which is sent to the clock recovery unit. The recovered clock is used to synchronize the decoder's clock or the STC. For analysis purposes, consider an elementary stream n, which has been successfully demultiplexed from the input FlexMux stream.

3.2 Buffers

For each of the elementary stream, the model has a decoder buffer B_T. B_T is sum of three smaller buffers that include a FlexMux buffer, a packetized header buffer and an elementary decoder buffer.

$B_T = B_F + B_m + B_d$

The decoder buffer of the nth stream is labeled as $B_T(n)$.

Denote the size of each of these buffers with SB_T, SB_F, SB_m, SB_d respectively.

Once the coded object is decoded, there may be a need for composition, before presentation. Hence these decoded objects are directed to composition memory units that are connected to each individual buffer. Assume there are M composition memory units ($C_0..C_{M-1}$) for each individual decoder buffer. Note M is an arbitrary number.

3.3 Timing Model

The decoder's clock is referred to as the STC (System Time Clock) and serves as the basis for all events timing model, within the STD, Figure 4. The ith byte of X enters the decoder buffer B_T at time $t = i$. The demultiplexor strips off the FlexMux header and sends the FlexMux payload instantaneously to the Buffer B_T. The header of the packetized elementary streams is also not sent to the decoder. Note, this depacketization is not explicitly shown in the figure 4.

This implies that the payload belonging to a certain object, n enters the Buffer $B_T(n)$ at time $t = i$. Furthermore, every access unit or AU,(coded representation of the object) is associated with a unique DTS and a CTS, i.e.

AU(p)→DTS(p),CTS(p).

If the timestamps are encoded in the bitstream, they are extracted at this stage. If the DTS and CTS are not coded, the decoder appropriately extrapolates the timestamps from the previously received ones. Referring to subsection 3.1, STC of the decoder has been synchronized with the first OCR at start up. STC is matched with all incoming OCRs with time.

Assume OCR(k) was the last received OCR at byte offset k or equivalently at time $t = t(k)$.

Also, Let R is the instantaneous bitrate of the FlexMux Stream in bytes per second. Let F be the frequency of the STC clock.

Then

$t(i)(\text{secs}) = \text{OCR}(k)/F + (i-k)/(R)$

In this hypothetical model, it assumed that AUs are decoded when the System Time Clock (STC), is equal to the DTS (time $t = t_d$) for that access unit. At that time the AUs are instantaneously removed from the Decoder buffers. This decode process creates composition units (CU), which are instantaneously placed in an available CU memory unit. When the STC is equal to CTS of a particular Composition Unit (time $t = t_c$), the CU memory unit which contains the CU under consideration, is considered available or empty. Further, for simplicity, if we assume that a CTS sorting procedure is performed at every STC clock tick, where the CU with the smallest CTS is always placed in the C_0 location, then C_0 is always considered available or empty.

3.4 Constraints

Overflow and underflow of decoder buffers can cause the loss of data and create unacceptable artifacts. To prevent this, the following constraints should be met in the STD.

- The Buffer B_T shall not overflow at any time.
- The Buffer B_T can underflow under certain conditions.
- The CU memory shall not overflow.
- The CU memory shall not underflow.

The total delay for each access unit between composition time t_c and STD entry $t(i)$ should be bounded. Here $t(i)$ refers to the time when the first byte of the access unit under consideration, enters the STD.

$t_c - t(i) < T$, where T is some non zero upper limit.

3.4 Conditions

The set of constraints in subsection 3.3 translates into a set of conditions, which must be satisfied by the encoder and the multiplexor. They are derived as follows.

Let the instantaneous buffer fullness of $B_T(n)$ at time $t = i$, be denoted as $S_nB(i)$. n denotes the Elementary(Object) number.

- Condition 1
  $S_nB(i)$ <= $SB_T(n)$, to prevent the Decoder Buffer overflow, for all i and all n.
- Condition 2
  To prevent a CU Buffer underflow, DTS(p) <= CTS(p), for all access units p.
- Condition 3
  To prevent a CU Memory Overflow
  MIN [CTS(k)] <= DTS(p), $k \in [p,p\text{-}M]$, where
  MIN[ ] represents minimum value, p and n represent access unit numbers. M is the total number of composition memory units available.

Example, Assume $M = 3$

Access Unit #	DTS	CTS	
0		0	c1
1		3	c2
2		5	c3
3	x	c4	

In this case, to prevent a CU Memory underflow,

$x >= \text{Min}[c1,c2,c3]$

The CU memory management does not hold meaning, if CTSs are not existent (natural video, natural audio etc). In this case, however, CTS is assumed to be equal to DTS (CTS=DTS).

The condition 3 then simplifies down to

DTS(k) > DTS(p), where k ,p refer to the access unit number and ($k > p$)

The sizes of the individual buffers shall dictate the Packetization/Interleaving strategy.

- $SB_F = k1.\ R_{MAX}(n)$, where $k1$ is a constant and ($0< k1 < 1$),
- $SB_M = k2.\ R_{MAX}(n)$, where $k2$ is a constant and ($0 < k2 < 1$)
- $SB_D = k3.\ R_{MAX}(n)$

Where $R_{MAX}(n)$ is the maximum bitrate for the nth elementary stream at a certain Profile/Level.

Profiles/Levels could also decide the total number of composition memory units, the parameter M.

4. SUMMARY

A summary of the proposed STD model for FlexMuxed MPEG-4 streams is presented. This model would be necessary to design multiplexors to create compliant streams that can play on compliant decoders. One key difference between MPEG-4 and MPEG-2 STD models is the possibility of composition. In MPEG-2, the decoder emits a frame every frame period. Hence the frame buffer is assumed finite and does not figure in the STD model. However, in MPEG-4 case, there is the possibility that an object that is decoded at a certain time may be composited at a later time. Hence if the composition memory is finite as may be the case in a hardware decoder, there has to be relationship between DTS and CTS. This relationship is derived in this paper.

5. REFERENCES

[1] ISO/IEC 13818-1 IS System Specifications.
[2] ISO/IEC 14818-1 FDIS Specifications.