Usages of DASH for Rich Media Services

Cyril Concolato Jean Le Feuvre Romain Bouqueau
Telecom ParisTech; Institut Telecom; CNRS LTCI
46, rue Barrault
75634 PARIS CEDEX 13
{concolato, lefeuvre, bouqueau}@telecom-paristech.fr

ABSTRACT
In recent years, audio-visual distribution over Internet has witnessed the growing usage of HTTP based delivery systems. While these systems have their drawbacks for some use-cases, they also have many advantages, the most important one being reusing the existing delivery infrastructure such as HTTP servers, proxies and caches. The MPEG group has started the standardization of the Dynamic Adaptive Streaming over HTTP (DASH) of major transport formats, MPEG-2 TS and ISO Base Media File, and mostly focuses on audio, video and subtitle formats. We believe Rich Media services have a role to play in this landscape, as a presentation layer for the audio-visual content first, but also as a dedicated media in the DASH content for real-time media-synchronized interactive services. In this paper, we present a study on usages of DASH for Rich Media Services.

Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols – *Applications.* H.3.2 [**Information Storage and Retrieval**]: Information Storage – *file organization.* H.5.4 [**Information Interfaces and Presentation**]: Hypertext/Hypermedia – *architectures.*

General Terms
Design, Experimentation, Languages, Standardization.

Keywords
Adaptive Streaming, HTML 5, HTTP, Interactivity, MPEG, Rich Media.

1. INTRODUCTION
In the past few years, new types of multimedia devices have emerged, in particular smartphones, tablet PC or connected SetTop Boxes and TV sets. These new devices have brought new usages for the consumption of audio visual services. Indeed, users now want to view the same audiovisual service either at home, in front of their TV or on their tablet PC in their bedroom, or on the move, with 3G services. These changes imply that methods for the efficient delivery of audiovisual services to a large number of users and devices have to be found. The number of audiovisual providers has drastically increased with the upcoming of Web videos, and technologies such as broadcast, which assumes that a return channel is not available, or RTP in unicast or multicast, which assumes that UDP traffic is available, are no longer appropriate for all providers. Recent developments in the industry [1] or in standardization bodies such as 3GPP [2] or MPEG [3] have defined new technologies for streaming of audio-visual resources over HTTP.

In the meantime, users' expectations with IP-based applications have risen. They are now used to consume audiovisual services as part of larger rich media services providing additional resources with the content. The upcoming HTML 5 standard is a good example of how video content can be enriched with graphics, animations, textual meta-data or community services.

At the same time, content providers need to ensure that their meta-data can be tightly synchronized with the content, as is usually the case in traditional broadcast environment for DTV systems for various use cases such as advertisements, voting, or annotations.

In this paper, we are interested in the usage of DASH in this environment. DASH [3] stands for Dynamic Adaptive Streaming over HTTP and acts as a superset of existing streaming technologies defined by other standardization bodies. We study how DASH can be useful to rich media technologies and how rich media technologies can be useful to DASH.

The remainder of this paper is organized as follows. Section 2 presents several scenarios where DASH and rich media technologies coexist. Section 3 describes what the state-of-the-art work in this area can achieve. Section 4 presents our study on the use of DASH and our proposals to improve DASH or existing rich media technologies. Section 5 presents some implementation considerations. Finally, Section 6 concludes this paper and presents future work.

2. SCENARIOS
In this paper, rich media services are defined as services that feature multiple media elements (audio, video, images, text, graphics) organized in a specific 2D/3D manner, with some dynamicity either in the organization of the media elements or in the content of the media elements (e.g. changing text, animated graphics) and with an interactive part. Such dynamicity may need to be finely synchronized with media elements.

With that definition, we can envisage the following types of scenarios: some where the rich media service is static and does not need to be streamed per se; and some where the rich media data (e.g. graphics, layout) changes rapidly. In the first scenario presented in this paper, we consider the usage of DASHed media

MMSys'11, February 23–25, 2011, San Jose, California, USA.

in a rich media service. In the second one, we consider that the rich media data can be DASHed. The third scenario is a combined approach of the first two scenarios, where only the dynamic part of the rich media content is DASHed.

2.1 Controlling a session from a Web page

In this scenario, we consider that a presentation is available with multiple video streams, one for each different camera angle, multiple audio streams for each language, and multiple subtitles streams. We envisage that an author wants to use a Media Presentation Description (MPD), defined by DASH, within a rich media document, for example an HTML 5 page. This page features some controls to enable the selection of a camera angle, the selection of a sound track, the activation/deactivation and selection of a subtitle track, and the ability to seek into the media presentation time line.

2.2 Visual Web Radio

In this scenario, we envisage an extension of traditional Web radios. Indeed, Web radios deliver their audio content over HTTP using Ice cast, Shout cast or over similar protocols. The extension of Web Radio into Visual Web Radio consists in adding a visual (non video) component to the audio component. The visual component carries additional visual information associated with the audio component. Some information can be synchronized and change more or less rapidly (lyrics, news headlines, quiz items). Some information can be quite static for example the weather forecast, the stock quotes. In this scenario, we consider that the service is integrated, in the sense that there is only one application/one window handling the visual and audio parts, as this is easier to ensure synchronization.

Figure 1 - Visual component of a Web Visual Radio when the speaker speaks about President Obama

We can illustrate the concept of visual radio with the following figures. During the news report, when the speaker speaks about President Obama, the visual component will show Figure 1. When he reports about the weather, the visual component will show Figure 2. At any time, the user can interact on the left-side text to display some non-synchronized information (e.g. horoscope, program grid).

2.3 Interactive Service Design Switching

In this scenario, we consider a TV program consisting of audio and video streams delivered in DASH with an associated interactive application, for example an SVG file. In order to avoid overloading the bandwidth by resending the SVG content on a regular basis, as it would be done in traditional broadcast application, the SVG content is made available on a Web server and downloaded by each user upon connection to the TV channel.

Figure 2 - Visual component of a Web Visual Radio when the speaker reports about the weather

The broadcaster wants to synchronize parts of the interactive application with the audio-visual content, such as active speaker name and biography in a talk show, as seen in the previous scenario. This requires tight synchronization between the interactive application and the DASH streams.

Although the DASH session is well structured into pre-identified periods, each corresponding to a single show and its associated interactive application, the content provider wants to be able to handle unpredictable events in its live broadcast:

- Red Alert Codes triggered by government agencies, for which the broadcaster wants to present an interactive application allowing fast phone call dialing or emailing,
- Failure of the contribution network, in which case the broadcaster may have to switch to a different program and change the current interactive application.

For these reasons, the broadcaster needs to be able to switch at any time between interactive programs during the lifetime of the DASH session.

3. RELATED WORKS

The WhatWG has started drafting some thoughts around the usage of Adaptive Streaming technologies in HTML5 [4], using the M3U8 format [1]. Their work is focusing on the control of quality of service at the player, including video layer switching or buffer management. Specific JavaScript APIs are proposed to handle stream switching at the browser level, based on various live parameters such as download rate, media buffer levels or frame drops. It does not take into account the possibility that a single session can be composed of many resources such as one video, several audio and several meta-data tracks (e.g. subtitles), all of them available in different resolutions, viewpoints and/or quality. Therefore, the proposed interaction with the underlying adaptive streaming subsystem is not sufficient to cover our scenario 1. The 3GPP group has also standardized a mapping between the MPD syntax and the SMIL syntax, but this syntax does not provide

ways for an upper presentation layer, such as an HTML <video> element, to select media representations based on user criteria such as language or viewpoint.

In the broadcast world, several technologies already offer the possibility to deliver interactive applications to the end user, including audio/video-synchronized events. Most are based on MPEG-2 Transport streams carrying either the application as files in object carrousels, such as in DVB-MHP [6] or HBBTV; or as rich media streams in MPEG-4 BIFS format, such as in T-DMB. Other technologies rely on an IP layer for transport [7], and carry the applications as a collection of files (using FLUTE) and data streams (using RTP) in SVG/3GPP DIMS format, such as DVB-H or ATSC-M/H. One key notion in all these systems is the carousel of media files. It allows users to download an application over a broadcast link in a reasonable time, but it is bandwidth costly. It also allows changing the complete application in all connected terminals if the broadcaster needs to.

In the Internet world, many technologies exist to allow dynamic modifications of a multimedia document such as HTML or SVG. We can classify these technologies in two categories. Pull techniques allow the client to request new data from the Web server; the most known example is AJAX-based technologies using the XmlHTTPRequest JavaScript object. Push techniques allow the server to send new data to the client. For a long time, these techniques were relying on script, either through progressive download or long-polling HTTP request from the client, and are often referred to as COMET [5] technology. Rich Media standards such as MPEG-4 BIFS, MPEG-4 LASeR or 3GPP DIMS, and more recently HTML 5 with its EventSource interface [8], have the ability to define data streams used by the server to send updates to the client. All these techniques are suitable for our Rich Media use cases, but we should keep in mind that AJAX, COMET or HTML5 techniques are heavier as they require scripting support even for simple operations such as text replacement and have no support for audio-video synchronization.

4. STUDY OF DASH FOR RICH MEDIA

4.1 Linking MPD from a Rich Media Document

In DASH, a given media resource as delivered by the network is called a representation. A representation can be a multiplexed MPEG-2 TS or ISO Base Media File, for example containing video and audio tracks, or can be a single-media file, for example containing only audio in a given language. Single-media representations can be assigned to a given group. A group is a set of alternate media representation, and a DASH player is expected to play at most one representation from each group; therefore, groups have to be carefully defined. The possible combinations can be restricted by defining subsets, for instance to disallow a particular audio track when playing the Director's cut version of a movie. For our scenario, it must be noted that only groups and representations have associated identifiers.

We consider in this paper that the MPD we want to manipulate is made of single-media representation, since multiple-media representations do not offer any flexibility regarding audio or subtitle track selection, as they are all part of the same representation. We also only consider the use case where the MPD file is the same for all users and is not negotiated based on user preferences: although this later case may offer a pre-selection of tracks, it does not solve the case of interactive switching between tracks unless a renegotiation for a new MPD occurs, which is precisely what we want to avoid.

One simple approach for media selection in DASH would be to identify representations to be played from the HTML content. However, this would force the DASH player to select one given representation and we will lose the capability of DASH to switch between representations when bandwidth conditions change. In other words, the HTML author would have to manage the stream switching logic in its content, which is a real burden and requires monitoring many real-time parameters of the network stack.

A better approach would be to identify groups, as each group constitutes the set of alternative representation for a media. However there are scenarios for which this is not sufficient. Let us consider the case of language selection for audio tracks. If all tracks are set in the same group, there is no way for the HTML document to select one particular language by using only the group id. One solution could be to group audio tracks by language, therefore defining as many groups as they are languages, and defining subsets in the MPD to avoid playing together two audio tracks. Each group can then be referenced through its ID in the HTML document. However, if the HTML document wants to switch the quality of selected audio track, the group id is no longer sufficient to identify the possible representations. Let us now consider the case of multiview video: a DASH session offers two views at two different resolutions, each of the videos being encoding at two bitrates. The HTML user interface needs to be able to select the view, and may need to switch the resolution when displaying the video at a lower resolution. The DASH player may not always know the target resolution of the video, e.g. when the video is used for visual effects in an HTML 5 Canvas object; only the HTML presentation layer may hint this. In such a case, either the MPD describes each representation as a group, restricting their combinations with subsets, or another selection mechanism should be used for HTML. If the HTML layer wants to select representations based on another criteria (e.g. frame rate), more groups would have to be defined and subsets and selection processing becomes cumbersome. One final consideration is that the HTML browser would need to be aware of group and representation identifiers, which makes designing a generic HTML UI for any MPD harder. We therefore propose to identify a sub-set of selection criteria from the MPD schema and expose them to the HTML layer. The set of criteria is presented in Table 1.

Not all of the various criteria defined in MPD have been exposed. We assume that bandwidth-related criteria are handled directly by the DASH player for adapting to bandwidth variation. We assume that in most cases, the DASH player will automatically handle the TrickMode criterion, used in slow/fast motion forward/backward, when the HTML object modifies the playback rate of the content; in other words the HTML Web author should not have to touch it. We however expose this parameter to allow non-linear viewing of video content in HTML, for example when displaying video key frames when moving the mouse over the media timeline bar. In such a case, the playback rate of the overlaid video is null (video is paused) but the DASH player has to be instructed to select a representation where random access is faster.

Table 1 – MPD Selection Criteria exposed in HTML

Criteria	HTML Selection Action
width, height	Specifies the desired video resolution
lang	Specifies the desired language
frameRate	Specifies the desired frame rate
qualityRanking	Specifies the desired quality
viewpoint	Specifies the name of the desired viewpoint
TrickMode	Specifies maximum playback rate desired for fast forward or rewind

Currently, the typical way to embed video/audio resources within a rich media document is through the use of dedicated elements and attributes such as the HTML 5 <video> or <audio> elements and the 'src' attribute, or the SVG <video> or <audio> element and the 'xlink:href' attribute, or the BIFS MovieTexture or AudioSource elements and the 'url' attribute. The use of DASHed media should not deviate from this current practice, but additional tools are needed to enable scenario 1. It should be possible for Web page authors to pass parameters to the DASH player, in an interoperable way. We can envisage three complementary methods to extend existing documents and achieve scenario 1.

The first method is to define fragment identifiers. Fragment identifiers are appended to the URL and have the advantage of being language independent. An example is as follows:

```
<video src="dash.mpd#viewpoint=1
&width=176&height=144">
```

The second method is to define specific attributes for the selection of a particular resource in the MPD. These attributes may be defined in the host language and namespace or in a DASH namespace. This is illustrated below.

```
<video src="dash.mpd">
  <track kind=subtitles src="dash.mpd"
   dash:qualityRanking="1" srclang="en"
   label="English">
</video>
```

The third method is to define an Application Programming Interface (API), typically, using a generic Interface Definition Language (IDL), which can then be mapped onto concrete languages such as JavaScript. Here we present a possible HTML5 API to create a new TimedTrack object from an MPD representation.

```
<script>
 var videoElement =
document.createElement('video');
var track =
createTrackFromDASH('subtitles','lang','
en');
videoElement.addTrack(track);
</script>
```

Given the novelty of DASH and HTML5, there are currently no available implementations of DASH players in an HTML5 environment, and it is quite hard to predict how close the HTML5 browser and the DASH player will be integrated. This makes it difficult to choose one of the proposed method rather another. Our preference goes to the first method, using fragment identifier, as it is language independent and may therefore be used in non-HTML5 environment such as CE-HTML-based, SVG-based or MPEG-4 BIFS. This approach avoids defining extensions in the host language (new attributes or elements).

4.2 Using MPD to carry Rich Media Services

When the techniques mentioned in Section 3 are used in a broadcast channel, they require sending the entire service has in a carousel on regular basis, but take advantage of the tight synchronization between audio-visual data and the service updates in the broadcast. When transposed in an on-demand context over HTTP, the service updates provided by these technologies are usually carried in a dedicated channel, for instance HTTP for AJAX or COMET based technologies, or RTP for BIFS or DIMS. This channel follows a different path than the audio-visual data presented in the service, and may even originate from a different server; this introduces important delays in service update acquisition, such as RTP buffer and transmission times or HTTP round-trip delay. Such solutions are therefore not suited for tight synchronization of service data with audio-visual data. In this section, we investigate how the service updates can be carried in DASH to guarantee this synchronization while being bandwidth-friendly. We will take the example of a BIFS service, but our approach is more generic and can be used for any timed data.

4.2.1 Example of a T-DMB Digital Radio Service

We take the example of a complex service, stored in an MP4 file, which has duration of 6:40 minutes and is composed of:

- An audio stream, here an AAC sound track
- An MPEG-4 BIFS stream, used to display a visual scene synchronized with the audio track. The scene uses text, graphics and images. The scene features a live screen and some non-live information like the last weather forecast, the last horoscope, or the EPG.
- An MPEG-ODF stream used to describe when images are used.
- 41 images, not displayed all at the same time.

The BIFS stream is a continuous stream, hence meaningful for DASH applications. It is a bit different from typical video stream, as it is made of only 69 access units, very sparse. The original sequence contains only one Random Access Point at time 0.

4.2.2 Using DASH to deliver the example content

Our first approach in using DASH for the transport of this interactive service is to embed BIFS media data in the DASH session. The DASH session uses ISO Base Media File Format as a container, and each track is setup to use track fragments. The segment duration is selected to be 10 seconds. The initialization segment of the session conforms to the DASH specification and only contains track declaration with no media data.

To enable random access in the presentation, we recreate a carousel at the beginning of each segment by generating random access points in BIFS and OD streams and reinserting images used during the segment duration. In our tests, the impact of this content modification on the BIFS bitrate is quite important as can be seen in Table 2. The image bitrate is of course much more

important but this is not really problematic as images could be moved outside the DASH session, as explained in next example.

Our second approach, which fits our third scenario, is to extract the static part of the presentation (of the BIFS stream) from the DASH session and only convey the scene modifications in the DASH session. This avoids carouseling most of the presentation, and therefore leads to a BIFS bitrate close to that of the original file with the initial random access point omitted. Images in the BIFS scene are referenced through HTTP links rather than using the OD framework, which allows simple image replacement in the DASH session by carrying only one link rather than the entire image. This also greatly reduces the bandwidth of the OD stream as can be seen in Table 2. This slight increase is due to the fact that image links to OD are replaced with image links to http resources, which are larger text strings.

Table 2 – Bit rate of a BIFS Visual Web Radio over DASH

	Average bit rate (kbps)	Peak bit rate (kbps)
Original bit stream	0.168	27
Dashed bit stream (Approach 1)	7	147
Dashed bit stream (Approach 2)	0.195	28
Dashed bit stream (external RAPs)	0.098	5

In summary, what we learn from this experiment is that delivering an interactive application, initially designed for broadcast and carousels, using DASH requires actual modification of the way the application is structured. Static elements and dynamic synchronized elements need to be delivered separately.

4.2.3 Optimizing DASH for Rich Media

The previous experiment showed that it is possible to use DASH to deliver continuous and dynamic interactive applications, but a problem remains. The requirement in the DASH specification that the initialization segment shall not contain any media data forces the content provider to insert the Rich Media service in a segment. Since users may start playing at any segment, this implies that the Rich Media service has to be repeated often in the segments, which is bandwidth costly as shown previously. If the Rich Media service structure is the same throughout the DASH session and only modifications to the content (text data changing, animation triggering…) are streamed in DASH, such repetition is awkward. The same remark applies for any meta-data format stored as media tracks in the file (XML, JPEG files…). We therefore propose to allow non-empty initialization segments in DASH to carry the static media data used in the DASH session in order to save bandwidth. This allows the content to be delivered with the original bitrate as described in Table 2.

One additional limitation we faced to fulfill our third scenario is that the ISO Base Media File Format does not provide many tools to reference external, remote resources that may change over time. We investigated using the capability of the file to define an external data source for the track using the DataReferenceBox, which may point to an HTTP URL; however we faced a new limitation in DASH that forces all data offsets to be relative to the start of the movie fragment. This makes external data references for tracks unusable in DASH. We propose to remove this constraint on relative data offset for track fragments using external data references, thereby allowing a client to fetch sample data on a given server outside of the DASHed file. This method gives us the result presented in Table 2, with a much lower bitrate for the DASHed part of the Rich Media service.

Finally, as explained in our third scenario, the Rich Media scene has to be sent, or at least signaled, on a regular basis. We explained that we reduce the size of this scene by placing its static content outside the DASH session, but still there remains a minimal scene. In scenario 3, we want the possibility to completely replace its content, as in a classical carousel. This is achieved by sending periodical random access points (RAP), for example, at the beginning of a segment. However, we must have the ability to signal that a RAP can be discarded in normal playback, otherwise a client will reload the scene at each RAP and loose all the current interactivity (user input, scripting context).

The SampleDependencyBox tool available in the file format is a good candidate for our needs. This tool allows signaling whether a media sample depends on another on, whether other samples depend on this one and is depended on and whether the sample has redundant. The notion of redundant coding depends on the coding type, and is not defined for meta-data or scene descriptions (BIFS, SVG). Our proposal is to define that a sample tagged as redundant, not depended on and not depending on other samples can be discarded if a RAP or another such redundant sample has already been processed. An alternative way could be to add a new flag in the track fragment header box indicating all samples in the fragment can be discarded under the same conditions.

This signaling may be redundant with existing features in some languages, such as the RefreshScene command in MPEG-4 LASeR. However, such signaling would require the client to first download the sample data and then discard it. Our proposal simplifies this process by saving some bandwidth and can be used with any meta-data streams.

Figure 3 illustrates the various cases enabled by our solution. The BIFS track in the DASH initialization segment uses two data references, one pointing to the DASH file itself, one referring to a Web server through HTTP. The figure shows that the DASH server and the BIFS random access server do not have to be at the same location. The first segment received in the DASH session contains a regular BIFS RAP conveying the interactive service associated with the audio. Its data is contained in the segment and the client must process it. The i^{th} segment is a simple service update. The j^{th} segment is a discardable BIFS RAP, which is ignored by connected clients and processed only by clients connecting at this time in the session. The data of this RAP is not included in the segment but made available on an HTTP server, thereby saving bandwidth in the DASH session even when inserting the RAP at high frequencies. The k^{th} segment is a regular BIFS RAP which must be processed by all clients, however the RAP data is stored on the server rather than in the segment. This covers the use case of our third scenario where the broadcaster decides to unexpectedly switch interactive services for all connected clients.

Our proposal is not restricted to BIFS, it may be used with other description languages such as LASeR or SVG; it may also be used

with AJAX-based solutions querying a meta-data stream in the DASH session, as used in DTV environments such as HBBTV.

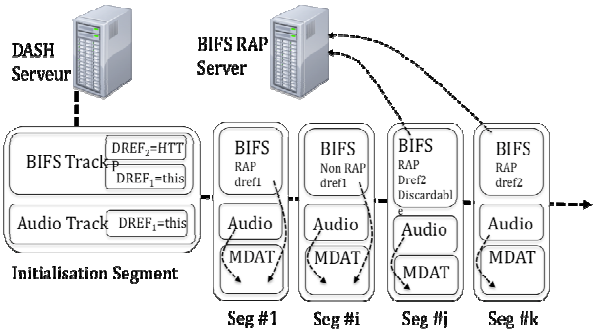

Figure 3 - Example of BIFS Carousel in DASH

5. IMPLEMENTATION

We have implemented our proposal in GPAC [9], an open-source multimedia framework supporting many interactive languages (BIFS, LASeR, SVG, X3D…) and delivery formats (MPEG-2 TS, ISO Base Media File…). Our implementation concerns both DASH media generation and playback.

We have extended the MP4Box tool to include a MP4/3GP fragmenter to generate segments of a given duration. The fragmenter may truncate the file at RAP boundaries to simplify content access in the DASHed file. The tool was also modified to generate random access points in BIFS and OD streams at a given frequency, in order to estimate the bandwidth overhead of using BIFS in DASHed without our proposed modifications.

We also have implemented DASH support in Osmo4, the GPAC player. A description of the implementation is given in Figure 4. We have validated support of both live sources using MPEG-2 TS as the DASH transport format, and on-demand sources using ISO Base Media File as the transport format. It supports both M3U8 formats and a subset of MPD files as defined in DASH.

In our implementation, the DASH player is in charge of parsing the MPD or M3U8 fetched from the server. It then selects the representation based on bandwidth criteria and other parameters set by the browser through the Selection API. The current implementation uses fragment identifiers in the MPD URL to configure the DASH player. The DASH player is then in charge of scheduling the segment downloads and sending them to the MPEG-2 TS or ISO File readers. We have extended our ISO reader to support external data references and redundant sample signaling for scene description.

Our implementation, released under the LGPL license, is available on the GPAC Web site http://gpac.sourceforge.net.

6. CONCLUSION

In this paper, we have investigated how Rich Media languages can be used with adaptive streaming over HTTP technologies, especially MPEG DASH. We have shown the need to provide standard ways to identify parts of a DASH session for track selection. We have also presented how Rich Media services can be carried in a DASH session along with audio and video data to ensure tight synchronization between the media data and the interactive service. Finally, we have proposed some light modifications to DASH and ISO Base Media File to allow building a smart, bandwidth-friendly data carousel of interactive services and meta-data in a DASH session.

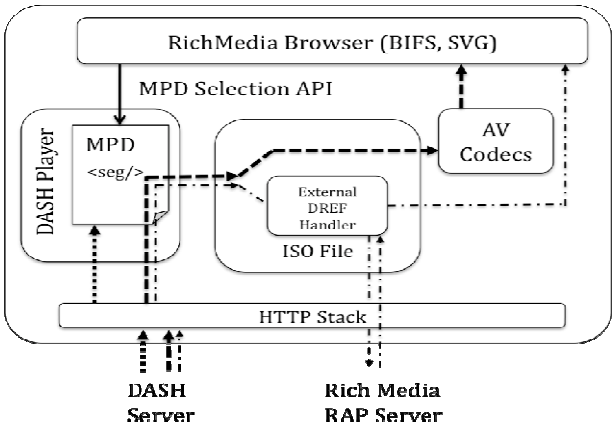

Figure 4 - DASH Implementation in GPAC

In future works, we will investigate implementation of real-time, live DASH services including interactive ones. We will evaluate how Rich Media adaptation to terminal characteristics and user preferences can be used in DASH, especially in live services.

7. ACKNOWLEDGMENTS

Part of the work presented in this paper has been funded by the French ANR projects Radio+ and HybRadio.

8. REFERENCES

[1] HTTP Live Streaming RFC, http://tools.ietf.org/html/draft-pantos-http-live-streaming-04

[2] 3GPP Adaptive HTTP Streaming, TS 26.234 http://www.3gpp.org/ftp/specs/html-info/26234.htm

[3] MPEG DASH, ISO/IEC 23001-6 CD (N11578)

[4] WhatWG Adaptive Streaming http://wiki.whatwg.org/wiki/Adaptive_Streaming

[5] E. Bozdag, A. Mesbah, A. van Deursen *"A Comparison of Push and Pull Techniques for AJAX"*, WSE 2007. 9th IEEE International Workshop on Web Site Evolution. DOI: http://dx.doi.org/10.1109/WSE.2007.4380239

[6] DVB DSM-CC Data Broadcasting, ETSI TR 101 202

[7] P.Leroux, V.Verstraete, F.De Turck, P.Demeester, *Synchronized Interactive Services for Mobile Devices over IPDC/DVB-H and UMTS*, 2nd IEEE/IFIP International Workshop on Broadband Convergence Networks, 2007. BcN '07. DOI= http://dx.doi.org/10.1109/BCN.2007.372743

[8] HTML 5 Server-sent Events, http://dev.w3.org/html5/eventsource/

[9] Le Feuvre, J., Concolato, C., and Moissinac, J. 2007. GPAC: open source multimedia framework. In *Proceedings of the 15th international Conference on Multimedia* (Augsburg, Germany, September 25 - 29, 2007). MULTIMEDIA '07. ACM, New York, NY, 1009-1012. DOI= http://doi.acm.org/10.1145/1291233.1291452