Je-LKS
Journal of e-Learning and Knowledge Society
The Italian e-Learning Association Journal
PEER REVIEWED PAPERS
FOCUS ON LEARNING THROUGH ENHANCED VIDEO
Vol. 11, n.2, 2015
ISSN: 1826-6223 | eISSN: 1971-8829

ADAPTIVE VIDEO STREAMING FOR TECHNOLOGY-ENHANCED LEARNING IN WORKPLACES

Maurizio Megliola[1]
Roberto Sanguini[2]
Michele Sesana[3]

[1] Piksel Ltd Italian Branch - Milano, Italy
maurizio.megliola@piksel.com
[2] AgustaWestland Training and Helicopter Support System Innovation Technology - Sesto Calende (VA)
roberto.sanguini@agustawestland.com
[3] TXT e-Solutions - Milano, Italy
michele.sesana@txtgroup.com

Keywords: technology enhanced learning (TEL), learning by doing, workplace, mobile learning, adaptive streaming

In recent years, the interest has been growing in video communication for training. Video clips have become an important learning resource, both within B2B distance training service offerings and the most popular social networking portals, such as YouTube™. In this scenario, however, the technical issue of how to proficiently deliver multimedia contents over heterogeneous wired or wireless networks remains a challenging aspect.
This is mostly related to the varying characteristics of the underlying network technologies, as well as the requirement to comply with multiple platforms and end-user devices (such as desktops, laptops, tablets and smartphones).

To provide a stable and uniform quality of experience to e-learners, the issue of monitoring the end-user and network context should be addressed and, leveraging on this information, the multimedia content should be dynamically adapted to match the device and network characteristics.
This paper introduces the Smart Multi-Channel Streaming Platform, a standard-based component of a novel operational framework developed within the research project TELL ME, with the aim to enable the context-based intelligent delivery of full multichannel streaming services for media contents to blue collar workers directly at workplaces. Thanks to the innovative use of HTTP adaptive streaming at workplaces, Learning Video quality adapts to the different Internet bandwidth and device capabilities, keeping a video playing and allowing the workers to successfully conclude their job having at its disposal the video as support. Next steps will include sensing the viewers and their surrounding environment to further optimize streaming.

1 Introduction

In recent years, there has been an impressive growth of the Internet video traffic. According to Cisco (Cisco, 2015, pp.1-3), in 2014 global mobile data traffic grew 69 percent and it was nearly 30 times the size of the entire global Internet in 2000, with almost half a billion mobile devices and connections added in 2014. In particular, mobile video traffic exceeded 50 percent of total mobile data traffic by the end of 2012 and grew to 55 percent by the end of 2014. Major global mobile data traffic projections and growth trends state that nearly three-fourths of the world's mobile data traffic will be video by 2019, increasing 13-fold between 2014 and 2019 and accounting for 72 percent of total mobile data traffic by the end of the forecast period.

In parallel, thanks to this greater worldwide access to the Internet and to the explosion of mobile phone users, there has been an increasing interest in the area of video for learning, with an actual appreciation for these technologies by end-users, as well as by teachers, opening up to rich media content as an important type of learning resource. In this respect, video may act as a powerful agent that adds value and enhances the quality of the learning experience.

The availability of new powerful devices, network connections and videos is exploited into the research project TELL ME[1], a EU co-funded initiative held by industry, universities, research labs, technology firms and innovation companies from all over Europe to improve training and work performance rapidly in small and medium-sized manufacturing environments, directly at the workplace, by using the latest technologies and insights. In this context the goal is to support the learning of blue collar workers (BCW) directly where the operations should be done and when it is needed. This approach moves the objective from "e-learning support on a PC" to "on the field" learning on full mobility, supplying learning instruction and supporting material to the workers while they are working, according to a novel learning operational reference

[1] http://www.tellme-ip.eu/

framework named eMeMO meta-methodology (Sesana *et al.*, 2014; Wild *et al.*, 2013). eMeMO, which takes its name from the steps it encompasses (enquire, Mix, experience, Match, and Optimize), defines a process by which the correct learning/training method for each specific worker/activity/context is selected, applied and monitored. Step 1 (enquire) provides instruments to express a learning need, to retrieve training resources that could help BCWs in filling gaps on e.g. new manufacturing processes and associated skills. Enquiries can be generated by BCWs, Process Managers, or semi-automatically (e.g. IoT-based wrong behavior detection).

Step 2 (Mix) helps in choosing an appropriate sub-set of the available training resources and feeding those resources into mixes that suit specific training needs. Learning mixes conveyed to the worker can contain any type of contents so that existing material can be reused, minimizing costs.

Step 3 (experience) provides the BCW with the actual training experience. Training resources (e.g. video clips) are delivered on multiple, complementary devices and can be contextualized on the fly according to the detected device, profile, environment, etc.

Step 4 (Match) uses engagement data from the training experiences to make predictions about trainee's state of competence, comparing learner behavior with the reference models specified at design time.

Step 5 (Optimize) allows for experimentation with new parameters for steps 2-4, seeking an increase in engagement and driving up performance, while reducing time-to-competence.

In this "training on the job" setting, the supply of video instruction and supporting material to the workers, provides additional challenges to the streaming of video content because the BCW can act in a big city with wireless or 4G connections, or being on countryside with limited connection (H+), or even being in a place where connection continuously changes bandwidth. The project has so provided the mandatory business need for an adaptive video streaming that, everywhere and in dynamic and sometimes difficult network conditions, should be streamed to the BCW. The worker must be able to somehow consume it in order to be able to successfully conclude the job. In fact, in these complex environments (e.g. aeronautical field maintenance), the timing for accessing the information is critical and a key indicator for the success of the system, while the quality should be just enough to understand the video.

This paper describes the Smart Multi-Channel Streaming Platform (SMSP), a standard-based component of the TELL ME system, which represents our solution to address the provision of full multi-channel streaming capabilities for media contents, including services for dynamically adapting the video quality to the available context (i.e. desired video quality matching the different network throughput and device power). This results in very little buffering, quick start time and a good experience for both high-end and low-end connections at workplaces, providing an innovative use of adaptive video streaming in "learning by doing" scenarios.

2 Technology Overview

In recent years the delivery of audio-visual content over the Hypertext Transfer Protocol (HTTP) got lot of attention and Internet streaming moved from an early concept into a dominant approach in commercial deployments. Its main characteristics are twofold: first, client consumption rate may be limited by real-time constraints as opposed to just bandwidth availability; second, server transmission rate (loosely or tightly) matches to client consumption rate.

Nevertheless, the quality of streaming experience, especially with regard to the delivery of video content to mobile devices, such as smartphones and tablets, can be frustrating due to wireless channel variations that, for example, lead to slow start-up, bandwidth fluctuation, and poor quality. Moreover, different kinds of video players, plugins, and network protocols may be required at the client side to run a given video clip. Further, mobile devices are already equivalent or superior to High Definition TeleVision (HDTV) sets in terms of graphics capabilities, often featuring high-density screens with 720p, 1080p, and even higher resolutions. They also come fitted out with powerful processors, making it possible to receive, decode and play HD-resolution videos. On the other hand, network and battery/power resources for mobile devices remain limited even with the introduction of 4G/LTE. Frequently users experience poor video quality due to re-buffering and inconsistent quality. All these elements led to technologies adapting to dynamic conditions anywhere on the path through the Internet, reducing bandwidth and power and improving quality in mobile video streaming, that is to the so-called Adaptive Bitrate Streaming (ABS).

ABS technology works by dynamically monitoring CPU and memory capacity and then making corresponding adjustments to video quality (Sodagar, 2011; Stockhammer, 2011). The core of the process includes encoding the original live or on-demand stream according to multiple qualities (bitrates), then segmenting each of the different bitrate streams into small parts and uploading them to appropriate web server folders. The segment length typically varies between 2 and 10 seconds. Segments are downloaded like traditional web objects, and a client can select bitrates for individual segments based on, for example, delivery bandwidth and playback horsepower, switching adaptively among those streams to deliver the optimal experience to the viewer.

The list of segments and bitrates is written into a manifest file. When the user accesses the multimedia file, the user's device processor requests the segments from the lowest bitrate stream (given in the manifest file). If the user's video player detects that the download speed exceeds the bitrate of the initial segment, it will request the next higher bitrate segment (again, written in the manifest file). This process will continue until a close match is found between the current bitrate segment and the user's available bandwidth. The video will then play at that bitrate. Later, if user bandwidth changes, it will request a different bitrate segment. Very often, the result is minimal buffering, quick video initialization, and a good experience on both high-bandwidth and low-bandwidth connections.

Differently from traditional streaming server-based technologies, such as Adobe's RTMP-based Dynamic Streaming, modern HTTP technologies have many advantages, such as NAT-friendliness and TCP's congestion avoidance, as well as the existing infrastructure's scalability using caches and content delivery networks (CDN). No server is needed to deliver the alternative streams, and the player is charged with both monitoring playback heuristics and retrieving the appropriate stream. Furthermore, it is supported by major industry actors and has been implemented in systems such as Apple's HTTP Live Streaming (HLS), Microsoft's Smooth Streaming (HSS), and Adobe's HTTP-based Dynamic Streaming (HDS) as their underlying delivery method.

HTTP Dynamic Streaming[2] was developed by Adobe. It enables both on-demand and live adaptive bitrate video transmittal of MP4 media over regular HTTP connections. HDS is an open-format solution that allows online media publishers to leverage existing network and cache facilities to deliver media to the Adobe Flash Platform with high efficiency. As with the other types of ABS, HDS can dynamically adjust movie playback quality to match the available speed of wired or wireless networks.

Microsoft has developed its own adaptive bitrate format called HTTP Smooth Streaming[3], as a part of IIS Media Services. It enables streaming media to Silverlight and other clients over HTTP. The format specification is based on the ISO base media file format and standardized by Microsoft as the Protected Interoperable File Format. HSS detects local bandwidth and CPU conditions to switch bitrates in near real time to offer the highest quality video that network and device conditions will allow. The files are delivered in a fragmented MP4 format and are stored as ISMV files.

Apple's solution to adaptive bitrate delivery, implemented by Apple Inc. as part of their QuickTime, Safari, OS X, and iOS software, is called HTTP Live Streaming[4]. It works by breaking the overall stream, packaged in an MPEG-2 Transport Stream, into a sequence of small HTTP-based file downloads, each download loading one short chunk (.TS files) of an overall potentially un-

[2] http://www.adobe.com/it/products/hds-dynamic-streaming.html

[3] http://www.iis.net/downloads/microsoft/smooth-streaming

[4] https://developer.apple.com/streaming/

bounded transport stream. As the stream is played, the client may select from a number of different alternate streams containing the same material encoded at a variety of data rates, allowing for seamless transitioning between bitrates. At the start of the streaming session, it downloads an extended M3U playlist containing the metadata for the various sub-streams which are available. This format is ideal for streaming video to iOS devices as it is supported natively on iOS 3.0 and later as well as on Safari 4.0 or later.

Each of the above ABS implementation uses different manifest and segment formats and therefore, to receive the content from each server, a device must support its corresponding proprietary client protocol.

A standard for Adaptive Bitrate HTTP Streaming of multimedia content has been developed by Moving Picture Expert Group (MPEG, responsible for many multimedia standards, including MPEG-2, MPEG-4, MPEG-7, MPEG-21, etc.) and published by ISO/IEC as International Standard in April 2012. It is known as Dynamic Adaptive Streaming over HTTP, shortly MPEG-DASH (Stockhammer, 2011) or simply DASH (ISO/IEC 23009-1). It represents the open source approach to ABS, with the aim of having a technology that is universally implemented compared to the more vendor-centric solutions listed above. It allows a standard-based client to stream content from any standard-based server, therewith enabling interoperability between servers and clients of different vendors. DASH is actually audio/video agnostic to the integration of different codecs and underlying application layer protocols, however the specification provides specific guidance and formats for use with two types of containers, the MPEG-4 file format or the MPEG-2 Transport Stream.

A comparison among HTTP-based Adaptive technologies is shown in the following table.

Table 1
COMPARISON AMONG HTTP-BASED ADAPTIVE TECHNOLOGIES

	Adobe HDS	Apple HLS	Microsoft SS	MPEG DASH
Manifest File	.F4M	.M3U8	.ISM	.MPD
Content File	.F4F	.TS	.ISMV	Media Presentation
Source Video Codecs	H.264, VP6	H.264, VC-1	H.264	H.264 + others (agnostic)
Source Audio Codecs	AAC, MP3	AAC, WMA	AAC, MP3	AAC + others (agnostic)
Segment Format	MP4 Fragments	MP4 Fragments	MPEG-2 TS	MP4 Fragments + MPEG-2 TS

	Adobe HDS	Apple HLS	Microsoft SS	MPEG DASH
File storage	Contiguous	Contiguous	Individual file per segment (pre iOS 5.0)	Contiguous or individual files per segment
Audio/Video/ Text packaging	Multiplexed in 1 Segment	Multiplexed in 1 segment	Multiplexed in 1 segment (pre iOS 5.0)	Multiplexed or separate segments for audio, video
Segmentation & Delivery	Adobe Interactive Server (or Adobe tools + Apache module for on-demand)	MS IIS (+ few other vendors)	Multiple vendors. Standard HTTP or Streaming servers	Multiple vendors. Standard HTTP or Streaming servers
Playback	Flash, Air	Silverlight	Apple iOS, Quick Time X	MPEG clients
Protection	Flash Access	PlayReady	AES-128 encryption	Flexible(e.g., OMA or UV)
Typical Segment Duration	2-4sec	2-4sec	10sec	Flexible

In general, to accomplish ABS, HTTP technologies require a common pair of files: an XML-based manifest file containing metadata describing the identity and location of the alternate streams (essentially telling the browser where the various pieces of media are located, but including also mimeType and other details such as byte-ranges), and content files specifically encoded and contained in media presentation files, so that the player can retrieve discrete blocks of the file during playback.

Adaptive streaming production implies two discrete analyses: how to choose the optimal number of streams and their configurations and how to customize the encoding parameters of the various streams to work within each adaptive streaming technology.

Basically, when choosing the number of streams, there are a few main factors to be considered:

- Identify the largest and the smallest resolutions we want to distribute
- Establish whether the use is entertainment-oriented (generally 4–11 streams) or educational or corporate (generally 2–5 streams)
- Consider the window sizes on your website (one viewing window requires 3-4 streams, while multiples should have at least one stream for each window size).

As far as the encoding, from a codec perspective, there are multiple options for some, but not all, adaptive technologies. For example, Apple's HLS is H.264 only, but Microsoft's Smooth Streaming works with VC1 and H.264, while

Adobe's Dynamic Streaming works with either VP6 or H.264.

As compared to either VP6 or VC1, H.264 delivers better quality at similar data rates, and it can be accelerated during playback by the GPU, enabling smoother playback on lower-power devices. H.264 also plays on most mobile platforms, so you can encode one set of streams for multiple targets.

3 The Smart Multi-Channel Streaming Platform

To support the eMeMO five steps, the TELL ME project has implemented a distributed architecture, spanning three distinct layers: a presentation layer holding multimodal user interfaces, a middleware layer including the core business logic components (implementing the eMeMO-powered functionalities), and a repository layer including distributed repositories and legacy systems. Across the three layers, a Virtual World Internet of Content (VW IoC) Platform has been developed (Sesana *et al.*, 2014). The goal of this component is to provide media ingestion facilities for acquiring new audiovisual contents (e.g.: through mobile devices) to be properly transcoded and stored within the TELL ME platform, making them available for further annotation through the media repository configuration interface provided. The Piksel Video Platform[5] is taken as the reference architecture for the VW IoC Platform, connecting to the different TELL ME media configuration tools such as Video Annotation and Segmentation Editor, Taxonomy Editor, etc.).

The above component has been recently enhanced with a Smart Multi-Channel Streaming Platform, which provides full multichannel streaming capabilities for media contents, including services for requesting the video content basing on the considered context, leveraging standard and open source Adaptive Bitrate Streaming technologies. Keeping in mind the project requirement of HTML5 multi-browser support, we adopted MPEG-DASH as common standard for ABS implementation. There are several key benefits in the adoption of the MPEG-DASH. Due to the fact that several major media companies took part in its development, the new protocol aims to eliminate technical issues in delivery and compression. In essence, it combines all of the technologies and standards into one, making streaming support seamless on all devices, as well as reducing transcoding costs. Content publishers can generate a single set of files for encoding and streaming that should be compatible with as many devices as possible, from mobile to Over the Top (OTT), as well as to the desktop via plug-ins or HTML5.

To play the content, the client first obtains a manifest called Media Presentation Description (MPD). The MPD can be delivered using HTTP, email, pen drive, broadcast, or other transports. By parsing the MPD, the client learns

[5] http://www.piksel.com/products/

about the program timing, media-content availability, media types, resolutions, minimum and maximum bandwidths, and the existence of various encoded alternatives of multimedia components, accessibility features and required digital rights management (DRM), media-component locations on the network, and other content characteristics. Using this information, the player selects the appropriate encoded alternative and starts streaming the content by fetching the segments using HTTP GET requests.

After appropriate buffering to allow for network throughput variations, the player continues fetching the subsequent segments and also monitors the network bandwidth fluctuations. Depending on its measurements, the player decides how to adapt to the available bandwidth by fetching segments of different alternatives (with lower or higher bitrates) to maintain an adequate buffer. The MPEG-DASH specification only defines the MPD and the segment formats, leaving the delivery of the MPD and the media-encoding formats containing the segments, as well as the client behavior for fetching, adaptation heuristics, and playing content, to the client scope. The figure below shows a simple streaming scenario between the Media Presentation Server and the DASH Client, where the elements in the scope of MPEG-DASH are encircled by the dotted line.

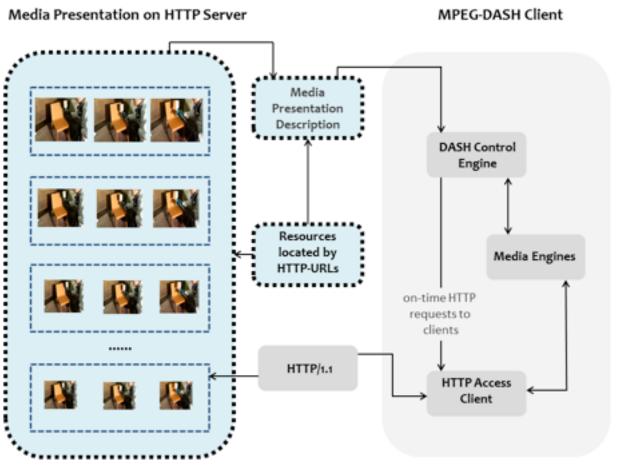
Media Presentation on HTTP Server
MPEG-DASH Client
Media Presentation Description
DASH Control Engine
Media Engines
Resources located by HTTP-URLs
on-time HTTP requests to clients
HTTP/1.1
HTTP Access Client
......

Fig.1 - Streaming scenario between Media Presentation HTTP server and DASH client.

In addition to defining formats for MPDs and segments, MPEG-DASH provides the ability to further restrict the applied formats by the definition of Profiles as defined on section 8 of ISO/IEC 23009-1. Profiles of DASH are defined to enable interoperability and the signaling of the use of features. Such a profile can also be understood as permission for DASH clients that implement the features required by the profile to process the Media Presentation.

In TELL ME, we adopted the DASH On-Demand profile (ISO/IEC 23009-1, 2012), which allows scalable and efficient use of HTTP servers and simplifies seamless switching. The primary constraints imposed by this profile are the requirement that each Representation is provided as a single Segment, that Sub-segments are aligned across Representations within an Adaptation Set and that Sub-segments must begin with Stream Access Points. In other words, you need provide a set of contiguous files and specify the bandwidth for each one and the appropriate file will be chosen automatically.

The On-Demand profile is identified by the URN:

```
"urn:mpeg:dash:profile:isoff-on-demand:2011".
```

To generate MPEG-DASH content we leveraged the MP4BOX content packager (GPAC, 2015) and x264 transcoder (VideoLAN, 2015). A video is given in some container format, with a certain codec, maybe also including one or more audio tracks. In order to describe the procedure followed, let's take as example a Learning Video provided by one of the three TELL ME end-users, keeping in mind that the following procedure has been followed for each video content supplied by all the end-user pilots. The file is called *aidima165.mpg*. To be prepared for MPEG-DASH playout, the video must first be ready for being used within segmented mp4 containers. To this aim, x264, an open source software library for encoding video streams, released under the terms of the GNU General Public License, has been used to transcode the video in an intermediate H.264/AVC format; the resulting file has then been segmented, by developing a content encoder on top of MP4BOX, which generates the DASH Media Presentation Description.

MP4BOX is a DASH packager available as part of GPAC, an Open Source multimedia framework which covers different aspects of multimedia, with a focus on presentation technologies (graphics, animation and interactivity) and on multimedia packaging formats such as MP4. It is useful for performing manipulations on multimedia files like AVI, MPG, TS, but mostly on ISO media files (e.g. MP4). Among its many scopes, it can be used for preparation of HTTP Adaptive Streaming content, precisely to generate media files conformant to various MPEG-DASH profiles, along with their corresponding MPD. The software does not encode or transcode media, and therefore requires alignment of stream access points among the different representations.

Back to our example, the following batch procedure summarizes the video H.264/AVC encoding process with the required properties:

```
   x264 --output aidima165.264 --fps 24 --preset slow --bitra-
te 2400 --vbv-maxrate 4800 --vbv-bufsize 9600 --min-keyint 48
```

```
--keyint 48 --scenecut 0 --no-scenecut --pass 1 --video-filter
"resize:width=1280,height=720" aidima165.mpg
```

All the command line parameters are explained in the following table.

Table 2
H.264/AVC VIDEO ENCODING COMMAND PARAMETERS

Parameter	Explanation
--output aidima165.264	Output filename. File extension is.264 as it is a raw H.264/AVC stream
--fps 24	Framerate which shall be used, here 24 frames per second
--preset slow	Presets can be used to tell x264 if it should try to be fast to enhance compression/quality
--bitrate 2400	Bitrate this representation should achieve in kbps
--vbv-maxrate 4800	Rule of thumb: set this value to the double of –bitrate
--vbv-bufsize 9600	Rule of thumb: set this value to the double of --vbv-maxrate
--keyint 96	Maximum interval between keyframes. This setting is important as we will later split the video into segments and at the beginning of each segment should be a keyframe. Therefore, --keyint should match the desired segment length in seconds mulitplied with the frame rate. Here: 4 seconds * 24 frames/seconds = 96 frames
--min-keyint 96	Minimum interval between keyframes. We achieve a constant segment length by setting minimum and maximum keyframe interval to the same value and furthermore by disabling scenecut detection with the --no-scenecut parameter
--no-scenecut	Completely disables adaptive keyframe decision
--pass 1	Only one pass encoding is used
--video-filter "resize:width=1280, height=720"	Can be omitted if the resolution should stay the same as in the source video)
aidima165.mpg	Source video

The next step is to add the previously created h264 raw video to an mp4 container as this is our container format of choice. The following batch command summarizes the above process:

```
MP4Box -add aidima165.264 -fps 24 aidima165_720.mp4
```

All the command line parameters are explained in the following table.

Table 3
MP4 VIDEO SEGMENTATION COMMAND PARAMETERS

Parameter	Explanation
aidima165.264	H.264/AVC raw video we want to put in a mp4
-fps 24	Framerate. H.264 doesn't provide meta information about the framerate so it's recommended to specify it. The number (in this example 24 frames per second) must match the framerate used in the x264 command
aidima165_720.mp4	Output file name

Afterwards, the segments and the corresponding MPD must be actually created. The following batch command summarizes the process:

```
   MP4Box -dash 4000 -frag 4000 -rap -segment-name segment_ ai-
dima165_720.mp4
```

All the command line parameters are explained in the following table.

Table 4
MPD SEGMENT CREATION COMMAND PARAMETERS

Parameter	Explanation
-dash 4000	Segments the given file into 4000ms chunks
-frag 4000	Creates subsegments within segments and the duration therefore must be longer than the duration given to -dash. By setting it to the same value, there will only one subsegment per segment
-rap	Forces segments to start random access points, i.e. keyframes. Segment duration may vary due to where keyframes are in the video
-segment-name segment_	The name of the segments. An increasing number and the file extension is added automatically. So in this case, the segments will be named like this: segment_1.m4s, segment_2.m4s, etc.
aidima165_720.mp4	The video we have created just before which should be segmented

The output is one video representation, in form of segments. Furthermore, there is one initialization segment, called *aidima165_dash.mp4*. Finally, there is a MPD, describing a manifest of the available content, its various alternatives, their URL addresses, and other characteristics. This step completes the DASH packaging of the source video.

Next step to finish the server-side SMSP process was to put the segments, the initialization segment, and the MPD onto a content delivery network, in order to serve multimedia content to end-users with high availability and high performance. In our case, we deployed the application in Apache Tomcat, an HTTP web server set up within the TELL ME VW IoC Platform, to serve.mpd files with mimeType set to "application/dash+xml". In general, the only requirement for a web server to support MPEG-DASH streaming is that it supports byte-range requests.

On the client-side of the platform, the Smart Player's configuration file must be pointed to the MPD on the web server, so to allow the content being ready to be adaptively enjoyed. In general, the ABS standard MPEG-DASH can be used in web browsers via the HTML5 Media Source Extensions (MSE) and JavaScript-based DASH players. Among the different players available, e.g. dash.js[6] of the DASH Industry Forum, or Bitdash[7], we upgraded the TELL ME Smart Player by building upon Video.js (Brightcove, 2015), an open source HTML5 JavaScript library for working with web video, vertically designed and developed in order to support the integration in an HTML5-based container, represented by the TELL ME entry point for the end-user. It integrates the logic needed to correctly estimate the dynamics of the available network throughput, control the filling degree of the client buffer in order to avoid playback interruptions, maximize the quality of the stream, while avoiding unnecessary video quality shifts, minimize the delay between the user's request and the start of the playback. It features three core parts: an embed code (Video for Everybody), a Javascript API (video.js) that works the same with HTML5, Flash, and other playback technologies, and a pure HTML/CSS skin (video-js.css) ensuring a consistent look between HTML5 browsers. It's licensed under the Apache License, Version 2.0.

Conclusions

Adaptive Bitrate Streaming designed to work efficiently over large distributed HTTP networks is quickly becoming the favorite way of streaming media over the Internet, enabling video systems to dynamically adjust playback quality, with very little buffering and quick start time, providing a good experience for both high-end and low-end connections. However, these technologies are remarkably more operationally complex than traditional streaming technologies. For example, they need additional storage and encoding costs. Nevertheless, HTTP-based ABS can leverage the same HTTP web servers used to deliver all other content over the Internet.

In this paper, we introduced the Smart Multi-Channel Streaming Platform, a standard-based component developed as part of the TELL ME novel pedagogical framework, which addresses the requirement of making Learning Videos always available to blue collar workers, independently of the workplace characteristics, allowing them to successfully conclude the job having at its disposal the video as support. Currently, the project is engaged in the final

[6] https://github.com/Dash-Industry-Forum/dash.js/wiki

[7] http://www.dash-player.com/

piloting, validation and evaluation phase by end-users in three different application scenarios, respectively 1) supplies and maintenance for the helicopter industry (access from hangars), 2) furniture making for luxury yachts (access from boats) and 3) quality inspection in textile industry (access from factory in the countryside). First results are satisfactory, allowing workers to experience Learning Videos from their workplace at the highest-possible quality playback by automatically detecting the user's network and playback conditions (which are subject to change) in real time, making it always available without lags.

Next steps will investigate about further streaming optimizations, for example by sensing the presence and proximity of viewers and their viewing conditions, taking into consideration attributes such as the distance of the eyes from the display, screen pixel density and ambient lighting conditions.

REFERENCES

Brightcove, Video.js, URL:http://www.videojs.com/ (accessed on 15th February 2015).

Cisco (2015), *Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update 2014–2019 White Paper*.

Evensen K., Kupka T., Riiser H., *et al.* (2014), *Adaptive Media Streaming to Mobile Devices: Challenges, Enhancements, and Recommendations*, Advances in Multimedia, vol. 2014, Article ID 805852, 21 pages, 2014. doi:10.1155/2014/805852.

GPAC, Multimedia Open Source Project, URL:gpac.io/downloads/ (accessed on 15th February 2015).

ISO/IEC 23009-1:2012, Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats, Cor.1 (available as w13495).

Sesana M, Scigliuto G, Zheng L, Genolini S, Civardi P, Bianchi S, Dongu M, Megliola M, *et al.* (2014), *TELL ME RW-DW-VW Platforms - Issue 2*, October 2014.

Sesana M., Naeve A., Marsh J., Wild F. (2014), *D1.3 TELL ME Methodology for Living Lab Learning, Life Long Learning*, Social Learning and TELL ME Methodology – Issue 2 TELL ME Consortium, July 2014.

Sodagar, I. (2011), *The MPEG-DASH Standard for Multimedia Streaming Over the Internet*, IEEE Multimedia, vol. 18, no. 4, pp. 62-67, Oct.-Dec. 2011.

Stockhammer T. (2011), *Dynamic Adaptive Streaming over HTTP-Design Priciples and Standards* In: MMSys '11: Proceedings of the second annual ACM conference on Multimedia systems New York, NY, USA: ACM Press, Feb 2011, S. 133-144.

VideoLAN, x264, URL:http://www.videolan.org/developers/x264.html (accessed on 15th February 2015).

Wild F., Scott P., Da Bormida G., Lefrère P., Naeve A., Isaksson E., Valdina A., Nina M., Marsh J. (2013), *Learning By Experience*: The TELL-ME Methodology (Phase 1) TELL ME Consortium, August 2013.