Dynamic ad-insertion and content orchestration workflows through manifest manipulation in HLS and MPEG-DASH

Robert Seeliger
Future Applications and Media
Fraunhofer Institute FOKUS
Berlin, Germany
robert.seeliger@fokus.fraunhofer.de

Daniel Silhavy
Future Applications and Media
Fraunhofer Institute FOKUS
Berlin, Germany
daniel.silhavy@fokus.fraunhofer.de

Dr. Stefan Arbanowski
Future Applications and Media
Fraunhofer Institute FOKUS
Berlin, Germany
stefan.arbanowski@fokus.fraunhofer.de

***Abstract*—The described proof-of-concept demonstrates, how manifest manipulation enables dynamic ad insertion and flexible over-the-top streaming workflows including ondemand-to-live services across a multitude of devices and platforms via MPEG-DASH, HLS and hybrid TV environments like HbbTV. Content manipulation and dynamic ad insertion is realized through non-video-intrusive technologies operating on manifest level in a highly flexible and scalable system architecture. APIs allow for individual stream management and channel orchestration. Dynamic digital video contents and advertising can be integrated per user and on the fly, in both linear and ondemand content. The evaluated system incorporates recent content stitching and content insertion techniques as DASH events, Xlink, SCTE35 and manifest rewriting for on demand and live sources. The different technologies have been evaluated against state-of-the art HLS and MPEG-DASH video players in multiscreen environments to derive best practices for short term utilization. In addition, recommendations for content creation and player implementations have been evaluated to further support the integration of those technologies in future solutions.**

***Keywords*—dynamic ad-insertion, HLS, DASH, adaptive bitrate streaming, Xlink, DASH events, internet delivery media**

I. Introduction

Usage of internet delivered video massively grew in the last years. There has been tremendous interest among consumers in watching movies on-demand and viewing video content on every device: television, video players, desktops, laptops, tablets, and smartphones. People enjoy the convenience of online video and the high-resolution and 360 features that are available nowadays. The fastest growth can be seen in the entertainment area, where video streaming portals and service offerings are the most relevant drivers for new technologies and content formats. But streaming is also moving beyond the entertainment area into education and health care, among other areas. Educators see documentaries, movies, and instructional videos as a vital part of the learning process. People love to learn through moving images as opposed to text alone and there have been tremendous advances in explaining complex topics through interactive video content [1]. Teachers are incorporating educational apps onto tablets and mobile phones, and helping students take responsibility for their own knowledge acquisition. All those new application areas for video content demand for more flexible, easy to distribute and interoperable video formats, overcoming the restrictions of traditional formats like RTMP, RTP or FLASH, which require specific network configurations and proprietary clients. Furthermore, they do only support one specific video bitrate with no means of adapting the current available network connection of the client.

Adaptive bitrate streaming formats (ABR) like Dynamic adaptive Streaming over HTTP (DASH) or HTTP Live Streaming (HLS) have become the streaming standard for the Web during the last couple of years and have harmonized video delivery across the internet. In parallel those technologies powered the massive growth of internet delivered media in total compared to traditional linear TV distribution via cable, satellite or over the air. As stated by the Cisco Visual Networking index (Cisco VNI), the global total Internet video traffic including business and consumer will be 80% of all Internet traffic in 2021, and was already up to 67% in 2016 [2]. Utilizing the HTTP protocol to deliver the media segments, ABR formats are perfectly integrating and scaling with content delivery networks as they are commonly used to ship IP traffic over the open internet. Thus, HTTP based streaming made massive video content delivery affordable and builds the foundation of the economic success, IP based video offerings gained the last years.

As a result of the video growth, content providers have created large libraries of video content that needs to be made available to customers and in a subsequent step, the content needs to be monetized in terms of subscription based models or by inserting advertisements. In addition, popular social media sites like YouTube, Facebook or Instagram have made video an integral part of their user experience and people are uploading a massive amount of video content every day. Besides simply adding advertisements, there is another way of creating valuable, targeted content from the large content libraries. Individual content assets, shows or social media clips can be orchestrated and enhanced with targeted or non-targeted advertisements to personalized, long-term content formats. These streams can be offered as on demand or scheduled linear streams following the concept of traditional TV programs.

This paper is structured as follows. Section II describes the technical foundations of adaptive bitrate streaming and the available ad insertion techniques in MPEG-DASH and HLS.

Section III deals with architecture and the main components of the FAMIUM DAI Service. Section IV gives a detailed overview of the ad-insertion process within the service. The document closes with an evaluation of the robustness and interoperability of the presented solutions.

II. Technical Foundations

A. Dynamic adaptive streaming over HTTP (MPEG-DASH)

Dynamic Adaptive Streaming over HTTP (MPEG-DASH) is an international standard for adaptive HTTP streaming which was published in April 2012 as ISO/IEC 23009-1. YouTube already offers 99% of its content via DASH [3] and also Netflix uses DASH as reference format whenever it is supported by the client platform. MPEG-DASH uses a manifest file, called Media Presentation Description (MPD), in order to provide a structured description of the media content. Each MPD file consists of one or more Periods used for insertion and logical segmentation of the content. The Periods have a start time relative to the start of the media presentation and consist of multiple Representations. Representations are alternative configurations of the media content and can differ in terms of the encoded bitrate, resolution, language or codec. Multiple Representations of the same content type like audio or video are grouped in Adaptation Sets. Each Representation consists of one initialization segment and multiple media segments. Whereas the initialization segment contains information for accessing the Representation, the media segments contain the actual media data [4].

To further promote and catalyze the adoption of MPEG-DASH and help transition it from a specification into real business, the DASH Industry Forum (DASH-IF) was formally incorporated in September 2012 [5]. Today it has more than 80 members including Microsoft, Netflix, Cisco, Google and Fraunhofer. One goal of the forum is to provide a free, open source DASH player which can be used as a JavaScript player or as a reference client to implement DASH players for other platforms. For this very reason, DASH-IF launched the dash.js project a web-based open source DASH player. Dash.js benefits from the W3C Media Source Extensions (MSE) and the Encrypted Media Extension (EME), two standardized APIs that are already present in most of the major browsers on the market. While the MSE extends the HTML Media Element to allow JavaScript to generate media streams for playback, the EME provides the functionality to control playback of protected content.

Based on the DASH-IF Interoperablity Points [6] Fraunhofer FOKUS has contributed different ad-insertion mechanism to the open source dash.js project. This includes support for DASH specific events, custom events, as well as XLink functionality. Events in DASH have a type, a timing and an optional payload and can either appear on inband level as ISO-BMFF boxes of type emsg or directly in the MPD as inline event. DASH specific events are defined in the DASH standard and are directly processed by a DASH client. They can be used to signal the client that the MPD needs to be reloaded, which is especially useful in a live streaming scenario, where new periods are added in order to splice ad content. DASH custom events are not directly processed by a DASH client. Instead they are passed from the client to an external application for further processing. They are the fundamental basis for a client based DASH ad-insertion architecture.

XLink allows the use of remote elements in the MPD. Those elements are not fully contained in the manifest file, instead they are resolved by the player. In order to use XLink, two attributes namely xlink:href and xlink:actuate have to be included inside an element. While the former is used to specify an URL to the remote content, the latter defines the resolution time. The resolution can be done either directly after the MPD has been parsed (xlink:actuate=onload), or at the time when the element is actually needed and processed (xlink:actuate=onRequest).

B. HTTP Live Streaming (HLS)

HTTP Live Streaming (HLS) is an adaptive streaming communications protocol created by Apple, which has become the de facto industry standard for adaptive streaming during the last years. As usual for adaptive streaming the source is encoded into multiple files at different bitrates and resolutions and divided into short chunks, usually having a length between 5-10 seconds. All chunks are referenced in the m3u8 manifest file, used in HLS to index the available content chunks. Up to 2016, HLS required content chunks multiplexed in MPEG TS (Transport Stream). The file format support was extend to fragmented mp4 with an announcement during Apples WWDC in 2016 [7], which made content chunks compatible with MPEG DASH segments. In contrast to MPEG DASH, HLS does not specify content periods or multiple period manifests, which could be used to introduce dynamic ad insertion through server side manifest manipulation by adding additional advertizing periods to the original content. HLS specifies a so called #EXT-X-DISCONTINUITY tag, which is used in the manifest file to decorate a generic discontinuity at the source. This could be a switch of e.g. encoding parameters, input sources, or an advertisement, which has been spliced into the source.

C. Hybrid Broadcast Broadband TV (HbbTV)

The Hybrid Broadcast Broadband TV standard was published in June 2010 by the European Telecommunication Standards Institute (ETSI) [8]. In November 2012, the version 1.2.1 has been approved and the elaboration of the second version is already in progress. HbbTV is based on different existing industry specifications, including the Open IPTV Forum (OIPF), the Consumer Electronics Association (CEA), the World Web Consortium (W3C), and the Digital Video Broadcast (DVB) standard [9]. The first revision adds standards from the International Organization of Standardization (ISO) and provides the features MPEG-DASH and MPEG-CENC [10]. The World Wide Web Consortium (W3C) is an international community, which has set itself the task of developing web standards. The aim of W3C is to define an open web platform that enables developers to build applications, available on any device, offering rich interactive experience. The consortium develops well-known technical standards, such as HTML, XHMTL and CSS, and represents the base of HbbTV. The Webbased Protocol and Framework for Remote User Interface on UPnP and the Internet (Web4CE) specification (CEA-2014) allows remote display of user interfaces, provided by third party Internet services on TVs, mobile phones, and other devices [11]. The ISO extends the HbbTV standard with two

different specifications: MPEG-DASH (ISO/IEC 23009-1) and MPEG-CENC (ISO/IEC 23001-7). The Dynamic Adaptive Streaming over HTTP (DASH) mainly specifies the dynamic adaptive streaming delivery of MPEG media over HTTP and a Media Presentation Description (MPD). Finally, HbbTV make recourse to the Digital Video Broadcast specification (ETSI TS 102 809) (see above) for the signaling and transport of applications and services in hybrid broadband broadcast environments.

III. Dynamic Ad Insertion and Content Orchestration Architecture

The FAMIUM DAI Service is an end-to-end solution for ad-insertion in MPEG-DASH and HLS. The system architecture is composed of multiple micro services, that can easily be deployed and integrate well with existing components and stream delivery infrastructures. Acting as a delivery middleware, the solution wraps the complexity of the ad-insertion process into easy to implement tools, which are manageable through an administration interface. It supports different ways of ad insertion, namely server-based and app-based ad insertion. The service covers the latest MPEG-DASH ad-insertion techniques like XLink and DASH inline and inband events. Moreover, it has support for ad signaling mechanisms like SCTE35 [12] and ad server standards like Video Ad Serving Template (VAST) [13]. The proof-of-concept implementation targets different platforms, including HTML5 browser, HbbTV 1.5/2.0 devices and native DASH and HLS player running on e.g. Android, iOS, FireTV or Chromecast. As shown in Figure 1, the system architecture comprises of four main components, accompanied by several 3rd party tools for optional functions. Those core components are namely the Aggregation Service, the Manifest Stitcher, a content management system (CMS) and the stream scheduler. Optional functions include an ad decisioning server, content transcoders and metadata services.

A. Aggregation Service (AGS)

The Aggregation Service (AGS) is the main component of the DAI and content stitching solution acting as aggregator for media feeds, content and ad assets. It runs the business logic to process stream requests, content conditioning and triggers manifest stitching. Furthermore, the AGS provides adapters to integrate with 3rd party services, ad servers and content management systems. It also handles and resolves the different ad-insertion technologies evaluated by this proof-ofconcept implementation.

B. Manifest Stitcher

The Manifest Stitcher performs individual manifest stitching, based on content playlists served by the Aggregation Service (AGS). The component creates MPEG-DASH and HLS manifests on an individual, per client basis on the fly. The Manifest Stitcher supports multiple manifest variations corresponding to use cases as static playlists, linear playlists, live playlists and mixed on demand and live input sources.

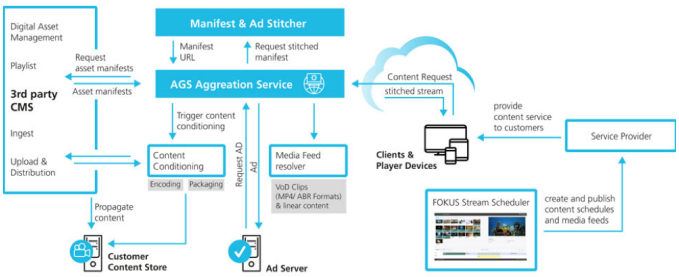

Fig. 1. Content and ad-insertion workflow

C. Content Management System (CMS)

The content management system stores and handles all content and ad assets, which are required for the service. It interfaces through a REST API with the AGS and provides support to encode and package media assets if needed.

D. Stream Scheduler

The stream scheduler (Figure 2) allows to orchestrate and serve content playlists created from on-demand and live sources and enrich them with advertisement pods or placement opportunities through e.g. a JSON based REST API or via an intuitive user interface. The solution supports static and live playlists for HLS and MPEG DASH that can be created from MRSS feeds, JSON playlists or by individual orchestration of content items through the user interface. For all types the solution implements the following content or ad-insertion methods:

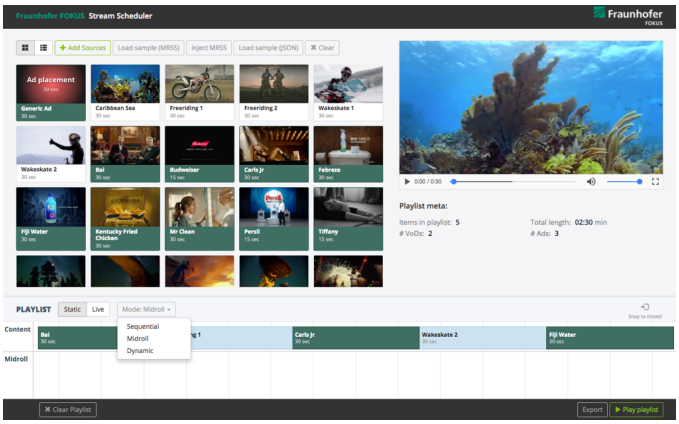

Fig. 2. Stream scheduler and orchestration tool

- Sequential: assets are stitched sequentially

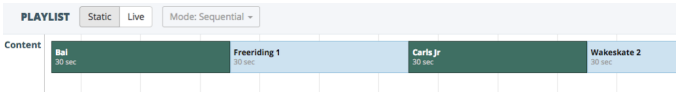

Fig. 3. Sequential mode

- Mid-Roll: assets can be splices into other assets like in traditional TV ad breaks

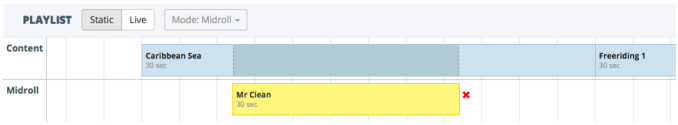

Fig. 4. Mid-Roll mode

Request sample payload from AGS to stitcher:

```
dash:[{    }, {    }]
0:{    }
baseUrl:"http://dai.com/content/dash/asset/"
inlineEvents:[]
mpd:"http://dai.com/content/dash/asset/asset.mpd"
subContent:[{mpd: "http://dai.com/content/dash/ad/ad
     .mpd",    }]
0:{mpd: "http://dai.com/content/dash/ad/ad.mpd",    }
baseUrl:"http://dai.com/content/dash/ad/"
mpd:"http://dai.com/content/dash/ad/ad.mpd"
spliceTime:12000
type:"ad"
type:"content"
1:{    }
baseUrl:"http://dai.com/content/dash/asset2/"
inlineEvents:[]
mpd:"http://dai.com/content/dash/asset2/asset2.mpd"
subContent:[]
type:"content"
```

- Dynamic: placement opportunities are stated in the manifest files and assets are requested and stitched during playback from e.g. an ad-server.

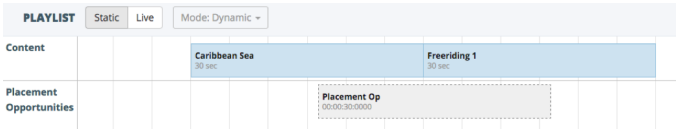

Fig. 5. Dynamic mode

The dynamic mode supports multiple ways of inserting content or ads into the stream during playback including server-side ad-insertion and client-side ad-insertion techniques. They are described more in detail in the upcoming section.

IV. Dynamic Content and Ad Insertion Techniques

Figure 6 shows a more detailed view of the architecture to insert content and ads through manifest manipulation using MPEG-DASH. In particular, we will focus on server-side adinsertion using multiple periods with and without XLink technology as well as client-based ad-insertion with DASH custom events.

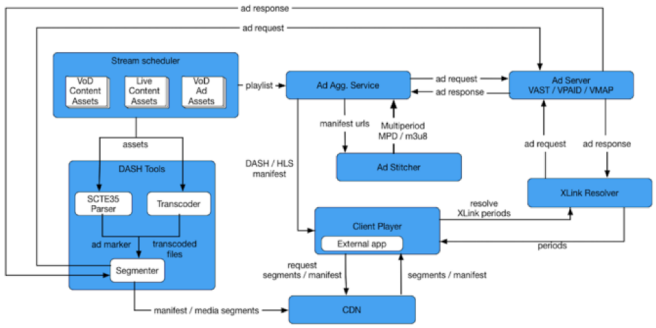

Fig. 6. Content and Ad-Insertion techniques for MPEG DASH

A. Server-side ad insertion

The easiest way to perform server-side ad-insertion is by using the multiple period approach of MPEG-DASH. Original content and content or ads to be inserted are described in the DASH manifest file as an individual period. During playback, the player transitions from one period to another period as defined in the manifest. The FAMIUM DAI solution supports multiple ways to insert ad or content periods in the manifest. The general workflow is depicted in Figure 7. The AGS component retrieves the original content (live or on demand) and advertisement cue points to stich ads into the original content. Based on this information, meaning where and what type of content needs to be stitched, the AGS creates a JSON based playlist file and sends it to the manifest stitcher, where the actual manifest manipulation takes place. The stitcher requests the manifest files for the original content assets as well as for the pre-conditioned ad asset and creates a new manifest comprising of multiple periods. This new manifest is send to the client and allows him to play the individual stitched content.

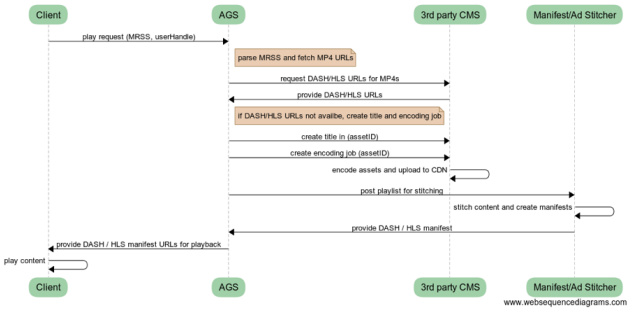

Fig. 7. Aggregation and Stitching sequence flow

A special implementation of the multi-period approach is by using XLink attributes. In that case, the actual content of the period is not part of the manifest but resolved on the client side during the playback process. The stream scheduler supports the insertion of XLink assets as follows:

1) The user defines a playlist in the stream scheduler UI and adds XLink content assets.
2) The Ad Agg. Service forwards the necessary information to the ad stitcher.
3) The ad stitcher inserts one or multiple XLink periods in the manifest file. The xlink:href attribute contains the URL to the XLink Resolver.
4) During playback the client player contacts the XLink resolver by issuing an HTTP GET request to the xlink:href url.
5) The XLink resolver contacts the Ad Server and gets an ad.
6) The Xlink resolver creates a valid ad period and returns it to the client player.
7) The client player merges the period back into the manifest and plays it.

Although multiple periods are standardized in MPEG DASH, we experienced in our evaluations, that the player support for this feature is still not widely available. Hence, we

decided to implement a more common method, the individual packaging.

The basic idea behind that approach is to ingest content or ads into DASH and HLS Streams by repackaging the stream. The output stream behaves like a linear stream, that does not require special player capabilities like multi period support. The re-packaged content can be handle by the player in the same way as the linear original stream. To guarantee a smooth playback without re-initialization of the video player, it is necessary to condition the original stream and the content to be inserted with the same content specifications in terms of audio and video coding settings. Usually this is done through preconditioned content assets or a on the fly transcoding of content and ads before re-packaging. The packager takes care of renumbering the stitched media segments and all subsequent following segments from the original content stream to ensure, that the outgoing stream can be handled like a traditional DASH live stream. This method guarantees the highest interoperability with players but requires more effort for the processing compared to client side ad-insertion or pure manifest stitching using multiple period manifests.

B. Client-side ad-insertion

With MPEG-DASH client-side ad-insertion is possible by inserting custom events in the manifest or the segements. Those types of events are not directly processed by a DASH client, instead they are passed to an external application. Typically, a DASH player API would allow an application to register for custom events. On occurrence of an event, a notification is thrown and the event is delegated to the application. The FAMIUM DAI Service supports client-side ad-insertion by using custom events in the following way:

1) The user defines a playlist in the stream scheduler UI and specifies all ad related information.
2) The Ad Agg. Service forwards the necessary information to the ad stitcher.
3) The ad stitcher inserts one or multiple Eventstream/Event elements in the manifest file.
4) During playback, the client player forwards the DASH custom events to an external application.
5) The external application contacts the Ad Server and gets an ad manifest/mp4.
6) When the presentation time of the ad is reached, the application pauses the main content and starts playing the ad in a second video element. Ideally the ad has been prebuffered and the event was signaled at least 4 seconds before the actual presentation time.

C. Integration of a VAST Ad Server

In real world scenarios the communication between clients and ad servers is based on an ad serving template like VAST. The FAMIUM DAI Service is able to interact with such servers on different levels. For server-side ad-insertion the AGS supports bidirectional communication with a VAST server. It acts as a middleware for the communication between stream scheduler and ad server. The input for the AGS can be either a specific number of single advertisements or a duration for the complete ad pod. It translates the input to a valid requests and parses the VAST response to a specific output format which can be interpreted by the manifest stitcher. The AGS is also capable of triggering encoding and packaging jobs for assets which do not exist in the target DASH or HLS format. Besides, the XLink resolver also communicates with a VAST ad server. In contrast to the AGS, the XLink resolver requires the assets to be already encoded and packaged in the right format. If the ad server returns mp4 files it will map those files to the prepackaged DASH assets. For client-side ad-insertion an external application handles all the VAST requests and responses. Since the client has full control over the playback it does not require the assets to be in DASH format, instead it can play the advertisements as plain mp4.

V. Evaluation and results

We evaluated the above described techniques for content orchestration and ad-insertion on multiple state-of-the-art DASH players including the DASH JS reference player dash.js version 2.5 and Android Exoplayer for native DASH support on Android powered devices. All evaluated techniques are able to support the integration of ad servers using e.g. VAST to fetch ads and allow for personalization of the content stream. Also, the support for SCTE35 is not limited to a specific method from the above-mentioned content orchestration and ad-insertion techniques. There is only a difference in how the integration needs to be done to allow content or ad-replacement based on SCTE35 triggers in the content stream. In terms of scalability and robustness against ad-blocker solutions we identified differences, that are described below.

A. Manifest manipulation using multiple periods

This technique scales best compared to the other options. The reason is, that manifest manipulation does operate on manifest level only. There is no need to encode or re-package the content assets and streams as long as they are conditioned all following the same content specs. This is usually handled by the Aggregation Services with support of 3rd party content management systems or direct connected transcoder and packager components. Thus, manifest manipulation using multiple periods is the most efficient way to create personalized ABR content. As multiple periods are used for both content periods and ad periods, ad-blocking solutions are not able to detect inserted ads. This solution will also be supported by HbbTV 2.0 capable TV sets.

B. Individual packaging using single period

Individual packaging is required for clients, that do not support the multiple period function for MPEG-DASH. Testing has shown, that typically the switch between periods is the most critical part for the players besides seeking in the content across periods. From a scalability perspective, this method is more complex and requires enhanced resources to perform real-time stitching, also called just-in-time-packaging. It offers a high grade of interoperability, because there is no enhanced player requirement to playback the re-packaged stream and it is the most robust technique to prevent ad blocking.

C. XLink

XLink requires client side support for XLink periods in the created manifest files. Ad-blockers might be able to

detect the requests to the XLink resolver and thus, block ad-replacements. The solution scales well, as there is no need for re-packaging or transcoding of the content.

D. Client side ad insertion using custom events

Client side ad insertion usually operates with two video objects in the client. Custom events in the manifest or the segments itself signal ad replacements to the client. The client switches between the original content in video element one to the alternative content or ad in video element two. This solution scales well, as all operations are performed on the client. Ad blocking is possible by preventing client requests to the ad server or hiding the second video element.

VI. Outlook

Dynamic ad insertion and dynamic content orchestration will be a core feature for future video distribution over the internet. The techniques will allow personalized and more targeted content streams for multiple application areas covering entertainment, TV, e-learning and more. The techniques evaluated in the proof-of-concept are currently not supported by the majority of available DASH players, but we have seen during the evaluation period, that especially multiple period support will be the most relevant feature for dynamic content insertion for both linear live and on demand streams. But even client-side ad-insertion techniques will be massively used in the future by e.g. IPTV operators or closed streaming environments, where client and server functionality can be coordinated through e.g. specific SDKs. Traditional broadcasters are moving more and more into Over-the-Top (OTT) streaming to distribute their channels to mobile and personal devices like smartphones and tablets. This will additionally boost the demand and utilization for dynamic content insertion techniques, because these new distribution channels will be bundled with content and advertising formats that are more individual for each viewer.

References

[1] R. Seeliger, C. Rck, and S. Arbanowski, "An approach for a cross-platform utilization of interactive content," in *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, Jan 2011, pp. 357–361.

[2] Cisco. (2015, Jul) Cisco Viual Networking Index (VNI). Accessed 2017.06.20. [Online]. Available: http://www.cisco.com/c/m/en_us/solutions/serviceprovider/vni-forecast-highlights.html

[3] N. Weil, "The State of MPEG-DASH Deployment," 2014. [Online]. Available: http://www.streamingmediaglobal.com/Articles/ReadArticle.aspx?ArticleID=96144&PageNum=3

[4] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles," in *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, ser. MMSys '11. New York, NY, USA: ACM, 2011, pp. 133–144. [Online]. Available: http://doi.acm.org/10.1145/1943552.1943572

[5] D. I. Forum, "DASH Industry Forum ," 2016. [Online]. Available: http://dashif.org/about/

[6] DASH-IF, "Guidelines for Implementation: DASH-IF Interoperability Points," 2016. [Online]. Available: http://dashif.org/wp-content/uploads/2016/12/DASH-IF-IOP-v4.0-clean.pdf

[7] J. Pantos, Roger; Schneider, "What's New in HTTP Live Streaming," 2016. [Online]. Available: http://devstreaming.apple.com/videos/wwdc/2016/504m956dgg4hlw2uez9/504/504_whats_new_in_http_live_streaming.pdf

[8] ETSI, "ETSI TS 102 796 V1.1.1 (2010-06) Hybrid Broadcast Broadband TV ," June 2010. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.01.01_60/ts_102796v010101p.pdf

[9] ——, "ETSI TS 102 796 V1.2.1 (2012-11) Hybrid Broadcast Broadband TV ," Nov 2012. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.02.01_60/ts_102796v010201p.pdf

[10] International Organization for Standardization, "ISO/IEC 23009-1 Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats," Tech. Rep., May 2014.

[11] CEA, "CEA-2014-A, (Including the August 2008 Errata) Web-based Protocol Framework for Remote User Interface on UPnP Networ ks and the Internet (Web4CE)."

[12] S. of Cable Telecommunications Engineers (SCTE), "SCTE STANDARD SCTE 35 2016 Digital Program Insertion Cueing Message for Cable ," 2016. [Online]. Available: http://www.scte.org/SCTEDocs/Standards/SCTE%2035%202016.pdf

[13] I. A. B. (IAB), "Video Ad Serving Template (VAST) VERSION 4.0," Jan 2016. [Online]. Available: http://www.iab.com/wp-content/uploads/2016/04/VAST4.0_Updated_April_2016.pdf