Personalized Dynamic Ad Insertion with MPEG DASH

Stefan Pham, Christopher Krauss, Daniel Silhavy, Stefan Arbanowski
Fraunhofer FOKUS
Kaiserin-Augusta-Alle 31
10589 Berlin, Germany
{stefan.pham, christopher.krauss, daniel.silhavy, stefan.arbanowski}@fokus.fraunhofer.de

Abstract— MPEG DASH enables OTT delivery of media content to almost any device. From a commercial point of view, dynamic ad insertion plays a crucial role. For broadcasters, content owners and advertisers new opportunities open up as advertisement can be personalized and delivered to any device. We show different approaches for selecting personalized advertisements via ad decision servers. Moreover, we evaluate existing ad insertion techniques and present solutions for interoperable ad insertion using MPEG DASH. We have extended the open source DASH Industry Forum's reference player "dash.js" with mechanisms for ad insertion. HTML5-based platforms can be targeted this way. We present different ad insertion workflows and highlight the flexibility that can be achieved with state-of-the-art technologies and standards.

Keywords—MPEG DASH, HTML5, ad insertion, dash.js, SCTE-35, XLink

I. INTRODUCTION

Content owners and broadcasters are increasingly using OTT (over-the-top)-delivery to reach a multitude of devices at any time and place. This area is as popular as never before with increasing customer numbers for OTT audio and video services worldwide. According to Cisco's Visual Networking Index [1], consumer internet video traffic is expected to be 80 percent of all consumer Internet traffic in 2019, up from 64 percent in 2014 and that Internet video to TV doubled in 2014. Internet video to TV will continue to grow at a rapid pace, increasing fourfold by 2019.

Requirements for OTT streaming solutions usually derive from existing solutions. These are on the one hand broadcast requirements, since users should ideally be given a comparable experience to what they are used to in broadcast TV in terms of video quality, playback and features such as subtitles, trick-play and alternative audio/video tracks. From a commercial point of view content protection, QoE (Quality of Experience) metrics and ad insertion are mandatory for many content providers.

Advertisement plays a key role in broadcast TV and is the most important source of funds for private broadcasters. Traditionally, TV ad insertion follows a one-to-many paradigm and is pre-scheduled by the individual broadcaster. Signaling of ad events and subsequent insertion of ad clips is spliced in at the broadcaster's site prior to the playout of the TV signal using, e.g. using SCTE-35 cue messages.

With the introduction of HTML5 Video a cross-device alternative to third-party browser plugins (e.g. Flash or Silverlight) emerged. The <video> and <audio> tag enable progressive media playback in the Web browser. In order to support adaptive bitrate (ABR) streaming another HTML5 extension is needed: The W3C Media Source Extension (MSE) specification is currently published as a Candidate Recommendation. It is well implemented across different browsers (see Table 1) and offers a JavaScript API to implement adaptive streaming, e.g. using MPEG DASH. The ISO standard Dynamic Adaptive Streaming over HTTP (MPEG DASH) defines media presentation and formats and is growing in popularity as well as becoming generally accepted by most vendors of pre-existing streaming technologies such as Microsoft Smooth Streaming, Apple HLS and Adobe HDS. OTT streaming services such as Netflix and YouTube have deployed MPEG DASH playouts. Furthermore, vendors of existing proprietary plugins like Adobe are now also moving towards HTML5-based solutions[2].

With Google's and Mozilla's decision to deprecate NPAPI (Netscape Plugin Application Programming Interface) – the interface that enables execution of plugins such as Flash or Silverlight – the relevance of the HTML5 video extension MSE increases. Content providers will have to transition to HTML5-based players soon in order to continue playback in Web browsers. Furthermore, features that have been in available in NPAPI plugins (e.g. ad insertion, trick-play, audio track switching) have to be enabled for HTML5-based players.

Section 2 provides details on relevant related works. In section 3 an overview of ad insertion technologies is given. This is complimented by an explanation of ad insertion workflows in section 4. Section 5 deals with personalization aspects in relation to the ad decision process. An evaluation of the "dash.js" client implementation is given in Section 6.

[1] http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html

[2] http://blogs.adobe.com/creativecloud/update-about-edge-tools-and-services/

Browser version	Media Source Extension
Microsoft IE 11 and Edge	✓
Microsoft IE <10	×
Mozilla Firefox > 41	✓
Google Chrome > 22	✓
Google Chrome (Android) 46	✓
Apple Safari > 7	✓
Apple Safari (iOS)	×
Opera > 14	✓
Opera Mini	×

Table 1 Support for W3C Media Source Extensions in current Web browsers

The DASH Industry Forum (DASH-IF) provides an open source DASH player called "dash.js" [3], which can be used as a HTML5-based JavaScript player. Dash.js takes advantage of W3C MSE and EME and adds a wide range of features including DRM, captioning and adaptation algorithms.

We see two main technologies, namely HTML5 and MPEG DASH as enablers for an interoperable and platform-agnostic streaming solution that fits all of these requirements. In this paper we will focus on the technical foundations and present different architectures for a personalized dynamic ad insertion solution. Based on the DASH-IF Interoperability Points we have contributed various ad insertion mechanisms to the open source dash.js project, which will be presented in this paper. These features allow the dash.js player to be compatible with both server-based as well as client-based ad insertion architectures as specified in the DASH-IF Interoperability Points.

II. Related Work

In this paper we do not introduce the technical foundations of MPEG DASH and assume that reader is knowledgeable of the basics. Basics of MPEG DASH are introduced by Stockhammer [1].

In 2013 we [2] conducted our first experiments with ad insertion using MPEG DASH. The solution proposed at that time was a proprietary client-based ad insertion solution. This non-MSE client would request ads as separate personalized MPDs, which are referenced in the MPD of the main content. In the meantime, W3C MSE, dash.js and DASH-IF Interoperability (IOP) guidelines have evolved to provide the means for an interoperable and dynamic ad insertion solution.

In 2015 we introduced a hybrid ad insertion [3] using MPEG DASH. Here we focused on the backend workflow to manage SCTE-35 cue messages, which were converted to broadcast-specific DVB events as well as broadband DASH events. This enabled client-side ad insertion into broadcast streams (HbbTV 1.0/1.5/2.0 compatible) and broadband (OTT using DASH), thus making it a hybrid ad insertion solution. In case of HbbTV the solution enables replacement of existing linear broadcast ads by overlaying them with full-screen IP-delivered ads [15].

Giladi et al. [4] present an overview of an OTT workflow in involving MPEG DASH that is based on the DASH-IF Interoperability Points. These are the same foundations that we consider in this paper. In addition, we focus on the client-side modifications to dash.js for an end-to-end ad insertion workflow.

Dubin et al. [5] propose a server-side ad insertion solution using MPEG DASH that does not require any modifications on the client side and further prevents ad blocking. The proposed solution introduces pointer URLs to media segments, which can be the main content or advertisement. Thus, additional server logic and load is introduced. Moreover, the effects on CDN caching and thus, scalability have to be considered.

III. Ad Insertion Technologies

In order to adapt to the specific needs of consumers in the rapidly developing market of video content, different committees have developed the following guidelines [6]: Video Ad Serving Template (VAST), Video Multiple Ads Playlist (VMAP), Video Player-Ad Interface Definition (VPAID), Video Ad Measurement Guidelines (VMAG), Digital Video Ad Format Guidelines and Best Practice, Digital Video In-Stream Ad Metrics Definition.

The Video Ad Serving Template (VAST) is a platform and device independent video ad serving template, which uses an universal XML schema to transfer advertising data from ad servers to video players. It works with Web pages, mobile devices and Smart TVs [6]. The standardized ad serving process defines two different main units, namely video players and ad servers. A VAST-compliant video player must be capable of handling VAST Responses and displaying ads as prescribed. It is accountable for the tracking and support of XML comments. Ad servers must be able to generate a VAST-compliant response after the request has reached them.

The Video Multiple Ad Request (VMAP) enables video content owners, who publish their contents through third-party video players, to keep possible advertising spaces under their own control. It means that they continue to be able to generate advertising sales. This standard allows content owners to define which kind of advertisement should be played during the video and its frequency. However, VMAP does not specify ad integration. This is what VAST is needed for [7]. The Video Player-Ad Interface Definition (VPAID) specifies an API, which enhances user experience through the use of VAST. The API provides the option to implement wide-ranging user interactions, which intensify the consumer's awareness. In this respect, VPAID is of particular interest to publishers and advertisers. In addition, they are able to gather precise values concerning the advertising impact [8]. The "Digital Video Ad Format Guidelines and Best Practices" aim to simplify the digital video advertising selling and buying process. These recommendations apply to the most current in-

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

stream ad formats in order to meet marketplace needs, such as the efficient delivery of ads and deployment of video players. Publishers can use these guidelines to self-attest their IAB compliance [9].

With the rapid expansion of OTT media, consumers are not able to consume all offered products at once any more – not even in a life time. Therefore, they have to find a selection of media items that they want to consume (item = the generalized term for ad or content in academic and data mining areas). An Ad Profiler helps its users (in this case the ad provider) to find a range of items the end customers might be interested in. Therefore, these engines use a set of different approaches to get a prediction of the users' behaviors. To find the best prediction for the current user, items and users have to be related to each other. Users can differ from other users more or less, depending on their properties. Items can differ from each other as well. Users and items are first and foremost related through feedback. A user can provide feedback automatically for instance when he watches a movie or manually when he rates the movie watched.

Taking the law compliance as given, broadcasters must collect usage data to predict the users' behavior and their preferences to find the best fitting content. One method to get the needed data is the collection of automatically generated feedback. Users can be tracked while watching the streams. Depending on the watched content, participating users have the choice between different streams, and common collaborative recommendation approaches can apply. So for instance the usage of association rules [10], known from phrases as "You watched program X and commercial Y, so you may also like commercial Z", can be used to predict suitable commercials. Another way is the collection of explicit feedback, as end customers can directly rate actual contents. Collaborative filtering algorithms are defined in order to predict ratings and to recommend the potentially best fitting commercials. The Slop One algorithm, developed by Yahoo [11], is qualified for predicting a user's rating by comparing all ratings of this user with the other users' rating of the according commercial.

IV. WORKFLOW

The DASH-IF IOP guidelines describe two different architectures for ad insertion, namely the server-based and the app-based architecture [12]. In the case of the server-based architecture, the ad selection process is triggered by a server component. For the app-based architecture it is the app or client.

Both server-based and app-based ad insertion require means to signal upcoming ad slots to the DASH packager and the MPD generator. The DASH-IF IOP guidelines recommend SCTE-35 cue messages for that purpose. SCTE-35 messages specify a technique for "carrying notification of upcoming splice points and other timing information" multiplexed into a transport stream [13]. Splice points are opportunities to either enter (In-point) or exit (Out-point) elementary streams and hence signal the start and the end of ad breaks in an MPEG2 transport stream.

A. Server-based Architecture

In a server-based architecture all the data and information needed to perform the insertion of the ad is concentrated in the DASH MPD and the segments. Single ads are expressed as a single period. For instance, in a classic midroll-scenario period one and period three are continuous content periods, while the period in the middle contains the ad content. Continuous periods can be marked with an optional asset identifier in order to allow the reuse of initialization data and DRM information across period boundaries [14].

The SCTE-35 cue messages are mapped to inband MPD Validity Expiration events, which trigger an MPD reload at the DASH client. At the same time the MPD generator creates a new MPD adding ad periods to the media presentation. As a result, the client player receives an updated MPD with additional ad periods. Besides, the use of remote elements allows a deferred resolution of the periods. Remote elements are elements that are not fully contained inside the manifest and are resolved via XML Linking Language (XLink). Only a subset of the XLink specification is needed for DASH. 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"). A deferred resolution should always be close to the playout time of the corresponding period and should leave the player with enough time to parse and interpret the resolved element. In the context of ad insertion, the use of remote elements is very helpful in order to dynamically decide which advertisement is shown to the viewer. As a result, three basic scenarios for the server-based insertion of the ad periods exist:

1. Immediate ad decision: At each client request for the MPD the generator will create a custom manifest depending on the user who is requesting the file. For that purpose, the generator will contact the ad decision server and include the ad periods in the manifest. The parts of the MPD that are the same for each user are kept in a template. Hence the generator only modifies the template on the arrival of a request and XLink is not needed.
2. Stateful cue translation: The generator keeps an MPD template and includes customized XLink URLs inside the ad periods, depending on information about the user. For instance, it might include a parameter specifying whether the person is male or female. On the client side the ad period will be resolved remotely via XLink.
3. Stateless cue translation: The generator keeps an MPD template and encodes all the SCTE-35 information into the XLink URL of the ad period.

Stateful and stateless cue translation require the resolution of the ad periods via XLink. The general workflow of the XLink resolution process is depicted in Figure 1. An HTTP GET request is issued using the URL of the xlink:href attribute. The

XLink resolver contacts an ad decision server with user specific parameters included in the request URL. The ad decision server responds with the ad content that should be presented. The XLink resolver creates one or more periods for that content and returns them to the DASH client. More on the ad decision process later on.

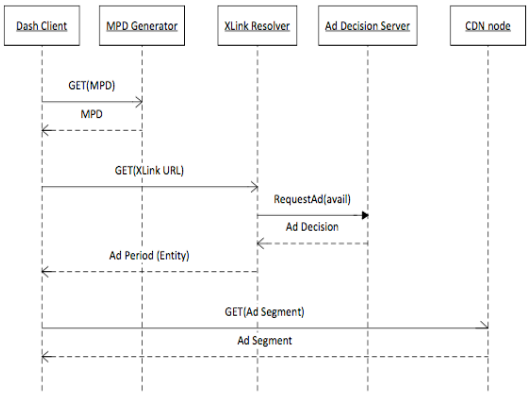

Figure 1 The XLink resolution process for the server-based ad insertion approach [10].

B. App-based Architecture

In the app-based architecture SCTE-35 cues are translated to application-specific inband events. During playback of the main content, the DASH client identifies the SCTE-35 inband events and hands them to the application logic. The application contacts the ad decision server through an interface like VAST in order to get the URL to an MPD that contains the ad content. This results in two different manifest files, one for the main content and one for the ad content. Hence, the use of multiple periods is not required. When it is time to play the ad, the app pauses the playback of the main content and starts the ad content on a second DASH client. After the ad is finished the client resumes the playback of the main content. Since two different DASH clients are involved, the playback typically also requires two video elements.

V. AD DECISION PROCESS

One of the biggest opportunities, but also biggest challenges, of personalized dynamic ad insertion is the selection of appropriate ads for individual customers presented on unique terminals. Personalized ads support the user's ad acceptance, thus increase the interest in the advertised product. Therefore, information for differentiating costumers is needed. This information can be very general, for example only representing the whole audience of a content item – e.g. the target group may be defined as a specific gender at a specific age. This information can be allocated manually by the ad provider or automatically using big data analysis. Moreover, it is possible to use available information of the client device – e.g. the region identified by analyzing the GPS location or IP address of a device. However, the best fitting item can only be selected, when getting individual data on single end customers and their viewing habits – from demographic information up to very individual behavioral data. In an all-encompassing ad decision infrastructure, the user himself can decide, which information he wants to provide. In general: the more personal data can be provided, the better the ad can be personalized. Depending on the detail level, the client application needs to mine different amounts of data, so the ad decision process can be more accurate.

The personal data can be directly transferred to the ad decision server or stored in the local storage of the application and only parts of the user profiles are transferred on demand. The latter is more transparent for the end customer, as he can edit and delete the data at every time. For instance, in a server based ad insertion process the available information on the user can be included as an XLink parameter inside the ad period. That way the client can transfer data like the gender of the viewer to the XLink resolver.

The manifest file for this scenario is structured as shown Table 2. It contains two content periods and one midroll ad period element with XLink attributes. The @xlink:href attribute contains an additional parameter "gender" which is used on the server side to return gender specific ad content. It is important to set the return value of the response to "text/plain" when returning multiple elements that are not wrapped inside a parent container. Otherwise the content is interpreted as non valid XML. In the final step the client merges the resolved content back into the MPD and starts with the playback. This approach is an example of a stateful cue translation.

```
<MPD>
 <Period start="PT0.00S" duration="PT600.6S"
id="movie period #1">
  <AssetIdentifier
schemeIdUri="urn:org:dashif:asset-id:2013"
value="md:cid:EIDR:10.5240%2f0EFB-02CD-126E-
8092-1E49-W">
</Period>
 <Period xlink:href="http://se-
mashup.fokus.fraunhofer.de:8080/mws15/
period_timescapes_komp?gender=male"
xlink:actuate="onLoad"></Period>
  <Period duration="PT600.6S" id="movie period
#2">
        <AssetIdentifier
schemeIdUri="urn:org:dashif:asset-id:2013"
value="md:cid:EIDR:10.5240%2f0EFB-02CD-126E-
8092-1E49-W">
</Period>
</MPD>
```

Table 2 DASH MPD with XLink sample

User profiling can be based on implicit feedback – without any user interaction. For instance, a TV application can only collect data on the watching behavior, e.g. when the user switched the channel and thus, skips an ad. Also regional data is very important, as service providers can select products for a specific region, as well. Explicit feedback, in contrast, is

more meaningful. If a user rates an ad as positive, bookmarks a product or clicks on the deeplink in order to buy that one, the items attributes will be concerned as user preference.

We trained and evaluated a server-side supervised algorithm based on both implicit and explicit feedback, called Support Vector Machine (SVM, [16]), that classifies ads for each user in "appropriate" or "not appropriate". Therefore, the user needs to provide a basic set of feedback data. In our case, the user needed to rate an ad on range *[1,10]*, where 1 is the worst and 10 is the best possible rating. The rating data is automatically transferred to the meta data attributes of that ad (109 features at all). As long as the user does not rate all ads with a 1, the ad decision server can select other appropriate ad items.

There are some evaluation metrics to determine the accuracy of ad predictions. The easiest way is to calculate the error of ad decision algorithms and filtering approaches by dividing the predicted value (if this ad was selected by the system for user u) by the real value (if the user liked this item). The Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) are calculated as

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N}$$

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} |p_i - q_i|^2}{N}}$$

$<p_i; q_i>$ is a prediction-real-value pair of N where p_i is the predicted value and q_i is the real value for item i. The lower the error, the better the algorithm. Formally, RMSE > MAE. The greater the difference between MAE and RMSE the more scattered errors are.

An evaluation with a group of 11 probands (aged 19 to 43; 4 females and 7 males) was made to analyze the accuracy of the whole system and to understand the users' imagination of such an ad recommendation system. Therefore, the test persons were asked to rate 20 different ads and give some personal information (gender and age). We used the 10-fold cross validation based on the RMSE for the evaluation. This cross validation type is repeated ten times. During every iteration the whole data set is split into another 10% of evaluation data and 90% of training data. The average value of errors defines the accuracy.

The results show that the Mean Absolute Error (MAE) is 1.98 and the Root Mean Squared Error (RMSE) is 3.77. The errors seem to be very scattered, because MAE and RMSE have a great deviation. The accuracy should increase when there are more users, items and ratings. The ad selection works very well for user 2 (average accuracy of 84.9%), user 6 (83.7%), user 10 (87.7%) and user 11 (87.1%). But user 9 brings trouble for the system, as the accuracy is about 62.7% on an average. The accuracies of the other users are in range of 78.0% to 81.0%. The total rating accuracy is 80.2%. So, when using our ad decision infrastructure 4 of 5 of all ads are classified correctly as of interest for the user or not.

VI. Evaluation of the DASH Player Implementation

In order to evaluate our implementation, we have tested the applications in different HTML5 browsers. Moreover we have extended the DASH-IF DASH Live Source Simulator[4], a tool to simulate DASH live streams, to supports MPD Validity Expiration events. The results of the test runs are depicted in Table 3 and Table 4.

Browser	dash.js 1.5 sample apps			
	Inline reload	Inband reload	XLink	Custom event SCTE-35
Chrome 45.0.2454.92	✓	✓	✓	✓
Safari 8.0.8	×	×	×	×
Firefox Nightly 43.0a1	✓	✓	✓	✓
Edge 20.10240	✓	✓	✓	✓
Opera 33.0.1990.115	✓	✓	✓	✓

Table 3 Support of the sample applications in different browsers

Browser	Live Simulator	
	Inband reload	Custom event SCTE-35
Chrome 45.0.2454.92	✓	✓
Safari 8.0.8	✓	✓
Firefox Nightly 43.0a1	×	×
Edge 20.10240	✓	✓
Opera 33.0.1190.115	✓	✓

Table 4 Support of the content originating from Live Simulator in different browsers

Google Chrome, Microsoft Edge and Opera are the browsers to support all of the test cases. For Chrome this fact is not surprising since Chrome is the platform, which was used for debugging during the implementation process. Mozilla Firefox supports all the sample applications, but has problems with the content originating from the Live Simulator for which the playback does not even start.

Based on these observations it can be concluded, that the

[4] https://github.com/Dash-Industry-Forum/dash-live-source-simulator

problems are related to the playback of multiperiod content. Nevertheless, further investigation has to be done to pinpoint the concrete source of the problem. The streams originating from Live Simulator have the same level of support with only Firefox having playback issues.

VII. Conclusion

Ad insertion is one of the last remaining challenges to be solved by content providers in order to move from browser-plugins such as Flash or Silverlight to platform-agnostic HTML5-based video players. We have integrated support for the server-based and app-based ad insertion workflows as specified in the DASH-IF IOP guidelines into the open source dash.js player.

Today already it is possible to enable personalized ad insertion using HTML5-based players and MPEG DASH across various browsers and platforms. With the finalization of the W3C media extension MSE specifications existing interoperability issues should vanish. Furthermore, we presented a solution for personalized ad selection based on Support Vector Machines. An evaluation showed a rating accuracy of 80% which means that 4 out of 5 dynamic ads recommended by our ad decision infrastructure are of interest for the audience.

References

[1] T. Stockhammer, “Dynamic Adaptive Streaming over HTTP – Design Principles and Standards”, in MMSys '11 Proceedings of the second annual ACM conference on Multimedia systems, New York, NY, USA, February 2011, pp. 133-144.

[2] S. Kaiser, S. Pham and S. Arbanowski. “MPEG-DASH enabling adaptive streaming with personalized commercial breaks and second screen scenarios”, in EuroITV’13 Proceedings of the 11th european conference on Interactive TV and video, New York, NY, USA, June 2013, pp. 63-66.

[3] R. Seeliger, D. Silhavy, S. Pham et al. “Cross-platform Ad-insertion Using HbbTV and MPEG-DASH”, in 2015 NAB Broadcast Enginnering Conference Proceedings, Las Vegas, NV, USA, April 2015, pp. 311-317.

[4] A. Giladi, Y. Syed, M. Iatrou et al. “Passing the Tuning Test: Providing Cable-Equivalent Ad-Supported Linear Programming Using MPEG DASH”, in IBC2015 Technical Papers, Amsterdam, Netherlands, September 2015.

[5] R. Dubin, A. Dvir, O. Hadar et al. “Novel Ad Insertion Technique for MPEG-DASH”, in Consumer Communications and Networking Conference (CCNC), 2015 12th Annual IEEE, Las Vegas, NV, USA, January 2015, pp. 582-587.

[6] Interactive Advertising Bureau (IAB): Video Ad Serving Template (VAST), Version 3.0. www.iab.net, 2012.

[7] Interactive Advertising Bureau (IAB): Video Multiple Ad Playlist (VMAP), Version 1.0. www.iab.net, 2012.

[8] Interactive Advertising Bureau (IAB): Video Player-Ad Interface Definition (VPAID), Version 1.0. www.iab.net, 2012.

[9] Interactive Advertising Bureau (IAB): IAB Releases New Standard Ad Unit Portfolio. http://www.iab.net, 2012.

[10] Linden, G.; Smith, B. & York, J. Amazon.com Recommendations - Item-to-item Collaborative Filtering IEEE Internet Computing, 2003, 76-80z

[11] Yu Kai, S. A.; Tresp, V.; Xu, X. & Kriegel, H.-P. Probabilistic Memory-Based Collaborative Filtering IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, 2004, VOL. 16, NO. 1, 56-69

[12] Dash Industry Forum, “Guidelines for Implementation: DASH-IF Interoperability Points Version 3.0,” Tech. Rep., Apr 2015. [Online]. Available:http://dashif.org/wp-content/uploads/2015/04/DASH-IF-IOP-v3.0.pdf

[13] Society of Cable Telecommunications Engineers, “ANSI/SCTE 35 2013,”Tech. Rep., 2013. [Online]. Available: http://www.scte.org/documents/pdf/Standards/Top%20Ten/ANSI_SCTE%2035%202013.pdf

[14] EBU and DVB, “TETSI TS 103 285: Digital Video Broadcasting (DVB); MPEG-DASH Profile for Transport of ISO BMFF Based DVB Services over IP BasedNetworks,” Tech. Rep., Mai 2015. [Online]. Available: http://www.etsi.org/deliver/etsi_ts/103200_103299/103285/01.01.01_60/ts_103285v010101p.pdf

[15] HbbTV Association, “HbbTV 2.0 Specification,” Tech. Rep., Feb 2015. [Online] Available: https://www.hbbtv.org/pages/about_hbbtv/HbbTV_specification_2_0.pdf Ltd., Natick, MA.

[16] Xu, Jin An, and Kenji Araki. "A SVM-based personal recommendation system for TV programs." Multi-Media Modelling Conference Proceedings, 2006 12th International. IEEE, 2006.