PASSING THE TUNING TEST: PROVIDING CABLE-EQUIVALENT AD-SUPPORTED LINEAR PROGRAMMING USING MPEG DASH

A. Giladi[1], Y. Syed[2], M. Iatrou[3], P. Navali[4]

InterDigital Communications[1], Comcast Cable[2], MediaExcel[3], Ericsson[4], USA

ABSTRACT

The OTT (over-the-top) model is experiencing rapid growth as a complementary video distribution channel. Operators and programmers are facing the challenge of providing the same quality of experience for both traditional and OTT viewers, and doing this at a scale.

Users are accustomed to robustness and quality of traditional video delivery ecosystems such as cable, IPTV, satellite and broadcast. A successful OTT deployment should be able to match, if not exceed, these.

In this paper we are providing an overview DASH implementation of robust ad-supported live linear services and an OTT-centric transcoder/packager architecture to support it. We discuss DASH features that make such service a reality, as well as a range of issues unique to such deployments – from content conversion to timing to scalability and fault tolerance, as well as additional benefits coming from tighter integration between intermediate transcoder-packager format and DASH.

Lastly this paper will discuss monitoring and verification of the end-to-end linear content delivery chain.

INTRODUCTION

An ever-growing share of US consumers uses OTT for viewing live content, such as sports events, while the vast majority is still served using traditional distribution methods, such as cable, satellite, IPTV and terrestrial broadcast. As an example, ESPN reported 17m traditional and 1.8m OTT viewers for the FIFA 2014 finals.

This is a good illustration of the reality faced by US programmers and operators: majority of viewers are still using the traditional decade-old content distribution mechanisms, while a rapidly growing OTT segment is served by adaptive streaming over HTTP.

Passing the tuning test – providing OTT customers with quality of experience (QoE) that is same or better than the traditional one is, thus, of paramount importance to the operators. This paper concentrates on a single use case: ad-supported linear channel.

Many of the issues that are irrelevant for on demand systems – such as "live edge" determination, end-to-end delays, schedule changes, coping with large audiences, etc., have a significant impact on viewers' QoE. It goes without saying that resilience and scalability are the basic requirement for any OTT deployment that offers premium linear content.

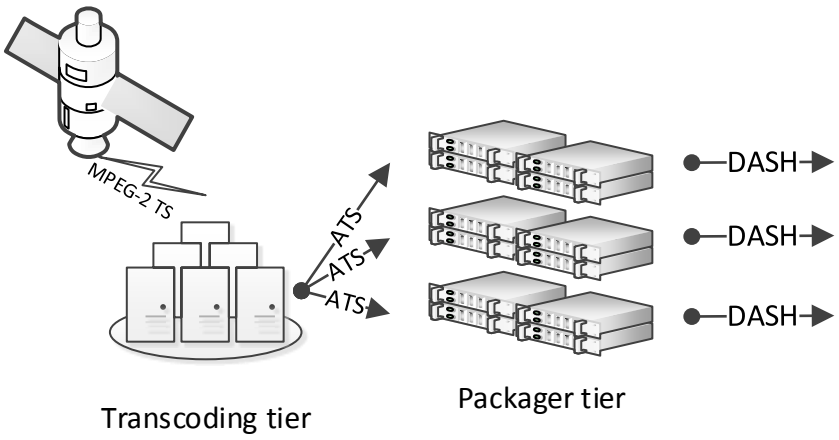

Figure 1 System Architecture

Our idealized system, depicted in Figure 1 can be summarized as follows: an MPEG-2 TS feed containing audio, video and SCTE 35 signalling is received and decrypted at the headend. It is then transcoded into multiple representations, some intended for OTT use and some – for traditional. The result is then packaged and encrypted for the intended crop of devices and mode of distribution, with OTT service provided using MPEG DASH.

We will assume that the reader is familiar with the basics of MPEG DASH and transport streams (MPEG-2 TS). Introductions to DASH are available in multiple papers, such as [1], [2], and [3], while [5] provides an in-depth introduction to MPEG-2 TS.

ABR-CENTRIC CONTENT GENERATION WORKFLOW

The simplest approach to generate DASH content from a real-time MPEG-2 TS feed can be summarized as (1) transcoding the source feed into the appropriate set of profiles, (2) delivering the segments to the origin server, and (3) letting CDNs take care of the distribution to the client. This solution undoubtedly works, but does not necessarily possess robustness and scalability properties expected from traditional systems.

The transcoding step can be conceptually decomposed into demultiplexing, pre-filtering, proper transcoding of elementary streams and packaging the resulting bitstreams into DASH segments for OTT distribution or MPEG-2 transport streams suitable for traditional distribution. Computational complexity of packaging is orders of magnitude smaller than the transcoding.

In a large, geographically distributed network, where multiple edge nodes on a closed provisioned network interface either with the clients directly or egress into CDNs, keeping transcoding and packaging together is computationally and economically inefficient. Physical separation between packager and transcoder, enables centralized transcoding processing. The horizontal scaling characteristics of such a multi-tier system with growth of audience are also significantly better than these of a traditional all-in-one transcoder. This tier separation is not entirely free – it is achieved at the expense of having to simulcast all encoded representations of the content rather than just deliver the source to each transcoder.

Transcoder-packager separation results in an architecture where the transcoder outputs all the necessary representations in an intermediate format and simulcasts them within the

managed network. For US cable industry this intermediate format is ATS (adaptive MPEG-2 transport stream).

A traditional packager implementation would use random access point signalling in MPEG-2 TS packets as segment borders. However, *virtual segmentation* – explicit marking of segment boundaries within a continuous stream, applied for each content component in the multiplex results in a simpler, more elegant and more robust system.

ATS is a regular valid MPEG-2 TS, which carries additional markers – encoder boundary point (EBP) structures – at the start of each segment. In addition to identifying the segment border, an EBP structure may carry acquisition time, thus providing a mapping between UTC and the earliest presentation time of a segment. From the operational standpoint, EBP structures are typically inserted by the transcoder.

ATS-based packagers are agnostic to a specific adaptive streaming system – it is built to support several systems simultaneously. In this paper we will see how integration of ATS-based packagers and DASH creates a more robust system for live linear services using DASH. A longer ATS-centric discussion of packagers is provided in [4].

Currently experimentation with ATS-based packagers is done in the unencrypted domain. However, encrypted domain repackaging of MPEG-2 TS into ISO-BMFF is possible with Common Encryption for MPEG-2 TS.

DASH SERVICE CONFIGURATION

General approach

The latest set of interoperability points published by the DASH Industry Forum, DASH-IF IOP 3.0, defines Main Live Operation (MLO) [8] and uses DASH ISO-BMFF Common Profile **Error! Reference source not found.**, defined in the latest version of the MPEG DASH specification are the best starting points for designing a linear ad-supported OTT service.

DASH MLO client uses short ISO-BMFF segments and *template-based addressing* to generate URLs corresponding to these segments. It will parse the downloaded segment for *inband events* and timing information and will use them to trigger *MPD updates asynchronously* only when there is a newer MPD version available.

HLS systems use playlists – explicit lists of all segments that need to be downloaded. DASH goes beyond that and uses the fact that encoders use naming conventions to predict segment URLs. This means that thousands of derived from a single-line template, and, since URLs of future segments can be predicted, there is no need to constantly poll for a new MPD – a significant gain over HLS.

Another DASH innovation is events. Events are messages sent either in media segments (inband) as ISO-BMFF `emsg` boxes, or as an element within the MPD. Their timing is relative to the start of playback. Events are extensible – user-defined payloads are possible, however DASH client itself handles events such as requests for MPD updates.

MPD update typically requesting a new MPD and continuing playback using it. This can be done by simple HLS-style periodic polling, but causing these updates using inband events is a very significant efficiency improvement. Additional improvement is the use of HTTP conditional GET requests to deal with spurious MPD update requests.

Dynamic advertisement

DASH-IF describes two different architectures for advanced advertising: app-driven and server-driven [8]. The main difference is that app-driven client uses non-DASH tools to communicate directly with an ad server, while server-driven client is a generic DASH client and ad server communication is hidden behind an HTTP proxy. MLO clients support both.

App-driven ad insertion relies on the client application to perform all communication with ad decision servers and only uses DASH events to transport application-specific messages. DASH-IF uses SCTE 35 cue messages for the purpose. Events can be carried in either inband or in MPD. The latter option has more overhead as it requires an additional MPD request and should thus be avoided.

Server-driven ad insertion relies only on native DASH tools – *periods*, *XLink* and asynchronous MPD updates – to provide the same functionality.

Periods are independent parts of presentations. In our case, an ad is a single period, while main content is a collection of continuous periods. In this case ad periods would appear between content periods. In this case, content periods would have identical asset identifiers, which will instruct the client to treat them as a continuation of each other. Periods do not necessarily contain complete information on content – remote periods require XLink resolution to determine what the DASH client will play.

In remote period case XLink URL is just a URL of a part of an MPD – one or more Period elements. Period *resolution* (i.e., download and processing) can happen just prior to the playback time of the period. This concept is used for just-in-time ad decision – an ad break is a remote period that can be resolved into zero or more ad periods.

In a server-driven case an inband event inserted prior to the start of the ad slot will trigger an MPD update. At the same time, a new MPD including a (possibly) remote period will be generated. The client will request a new MPD and resolve the new remote period.

WORKFLOW CONSIDERATIONS

Encoding and transport stream issues

OTT distribution adds several additional constraints for encoders and packagers: firstly, segments need to be time-aligned, which in practice translates into alignment of IDR (instantaneous decoder refresh) pictures and closed GOPs (groups of pictures between two IDR frames) with GOP length never exceeding segment duration.

The media segment duration should be kept below 3 seconds in order to achieve shorter channel acquisition time, more precise and efficient seek performance, as well as lower end-to-end latency. Segments with sub-second duration lead to significant compression inefficiencies and quality degradation. The latency-coding efficiency trade-off that should be taken into account, depending on the specific constraints of the video delivery service.

Existing encoding profiles geared towards traditional MPEG-2 TS distribution can still stay the same and may be used in addition to OTT-only profiles. Integration of older profiles has its pitfalls – e.g. switching between different aspect ratios and between progressive and interlaced content will often result in glitches.

MPEG-2 system information, such as PMT (program map table), may change and care should be taken to monitor it – a significant change (video or/and audio characteristics,

closed captioning, etc.) warrants starting of a new period. If this happens prior to the transcoder, though, it may be operationally simpler to handle these changes there and not at the packager, and thus provide the latter with a clean and continuous input.

The complexities of a hybrid OTT-traditional distribution workflow can be minimized by the use of a universal packager that generates content for both delivery paths.

Accessibility

The typical accessibility features expected from a US cable channels are CEA 608 / 708 closed captioning and associated audio services. The source feed that is the starting point of our content generation workflow is expected to already contain them.

Closed captioning in MPEG-2 TS based systems is carried within the video bitstream, and can support services in multiple languages. The same method was adopted by DASH-IF [8] and SCTE [12] for carrying CEA 608 / 708 captions in ISO-BMFF segments.

Caption service metadata, such as service-language mapping, are embedded in the original feed, and the packager can directly translate this metadata into MPD-based signalling defined by DASH-IF and SCTE [11]. An emerging W3C specification uses this signalling for creating HTML5 text tracks.

Carriage of additional audio tracks for vision impaired and hearing impaired viewers are also an important requirement. The main and associated audio tracks can come already mixed into a single stream (“broadcast mix”) or as two separate audio streams (“receiver mix”) that will be mixed at the audio decoder. DASH allows signalling audio track roles and expressing inter-track dependence for receiver mix in the MPD. This signalling needs to be derived in the majority of cases from descriptors in the MPEG-2 TS feed.

SCTE 35

SCTE 35 cue messages [9] are ubiquitous in US practice. Their main use is announcing avails, time slots operators can use for advertisement. In case of app-driven DASH ad insertion, cue messages will be passed through into inband DASH events.

In the server-driven case, the cue message will be translated into an inband event which will trigger an MPD update. At the same time, MPD will be updated, and – if remote periods are used – the cue message can be embedded in the XLink URL and eventually used as a part of ad decision parameters. Content identifiers passed in cue messages should be translated into period-level asset identifiers in the MPD, although it may suffice to be able to differentiate between in-network and out-of-network content.

SCTE 35 is also used for purposes unrelated to advertisement. The more interesting use cases are announcing program borders and schedule changes, content identification and mapping feed timing to atomic clock.

Some of the schedule-related changes (such as end of program) should result in adding a new period to the MPD and triggering an asynchronous MPD update.

In many cases there are multiple cue messages for the same event, and some messages announce event cancelation. Not every cue message should be passed to the client, especially when an MPD fetch is an expected consequence.

TIMING CONSIDERATIONS

Clock synchronization

The MPEG-2 system has a conceptually simple timing model: there is a single governing clock, STC (System Time Clock), which is the media clock and it can be reconstructed at the receiver using PCR (Program Clock Reference) embedded in the media. In practice this means translating STC into receiver wall-clock time using a PCR and an absolute arrival time of a PCR-bearing transport stream packet at the receiver's network interface. This model directly translates into DASH ISO-BMFF segments – the `prft` box **Error! Reference source not found.** has a very similar functionality and can be derived from PCRs. An important difference between traditional and OTT models is bitrate constancy – while PCR rate is expected to be constant, consecutive `prft` values are only expected to be non-decreasing.

PCR is an abstract value that is unrelated to any absolute clock or to start of a program. Anchoring STC – providing a translation between absolute time and a PCR value – is essential if we need to combine resources coming from different sources, even from different transcoders. Anchoring can be addressed in several ways using tools such as TEMI descriptors [7], EBP structure [13][10], and SCTE 35 [9] time descriptor.

The adaptive streaming has a "pull model", which means that the segment availability time also depends on the wall clock time at which the DASH client issues an HTTP GET request for the segment. This GET request is not guaranteed to succeed – if the segment is not available at the time the request reached the HTTP server, the latter will respond with an error. A DASH segment has an availability window, starting from the "live edge" time of its generation till the end of its time shift buffer (for catch-up TV services).

In cases where the client needs to operate at the "live edge", downloading the latest available segment, it is essential to know the precise time (according to the HTTP server clock) at which the HTTP GET request for the segment is guaranteed to succeed. DASH allows precise calculation of the availability window through explicit signalling. DASH requires a common clock across the delivery chain. Transcoders, packagers and edge HTTP servers are expected to be synchronized to a single clock. User equipment clock can drift, and DASH client may have different time source (e.g., GPS vs NTP). For this reason DASH allows time synchronization, where clients can explicitly request server time prior to requesting segments.

Client buffering delay

The difference between the time the segment download is complete and the time is sent to the decoder depends solely on the client, as scheduling HTTP GET requests is a prerogative of the client.

It is possible to live on the edge – immediately decode a segment as it arrives. DASH allows even further latency reduction – download of partially available segments using HTTP chunked transfer mode. However, living on the edge has its risks – the client is far more susceptible to running out of segments due to network conditions and thus to rebuffering. In order to avoid this, clients can build a buffer of several segments. The amount of buffering is a client decision, and it is entirely possible that different clients will operate on different buffer sizes. These may result in a noticeable difference between the times the same event is seen on different devices. DASH provides a tool for synchronizing

presentation delay by explicitly suggesting the distance between the “live edge” and presentation time of an access unit.

Additional non-media delays can be accommodated by an increased buffer size. We will discuss these, as well as server-side techniques for their minimization, throughout the remainder of this section.

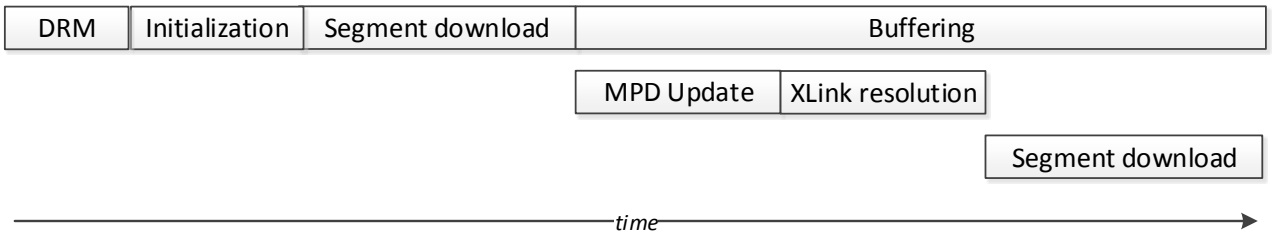

Figure 2: System delays

Period transition delay

The transition between different periods in a multi-period MPD has several implicit delays. The first delay that is encountered is the (re-)initialization delay for the decoder(s). This delay is unavoidable when transitioning between periods containing different content. In the case where periods are used for resynchronization or removal / addition of a representation, and the content is generated using the same encoding profiles, re-initialization is avoidable. DASH-IF, as well as an upcoming addition to MPEG DASH use explicit signalling to show that the periods are continuous and matching representations in the new period are just continuation of the ones in the previous period.

If the content is protected using Common Encryption, an additional delay would is necessary for license acquisition. In the worst case, the initialization segment would need to be downloaded in order to get information on license acquisition from the `pssh` boxes it contains. This delay can be avoided if the content of the `pssh` box is carried within the MPD. In this case DRM-related tasks can be performed in parallel with the download of the initialization and media segments.

Advertisement delays

The splicing delay is the delay between the time a notification of an upcoming change in content (e.g., an upcoming ad break) is received by the client and the time the client discovers the URL of the first segment of this new content.

In the app-driven case delays are outside the control of a DASH client as it passes the event it received to the application. The latter has to communicate with an ad decision server (e.g., a VAST server) and possibly start a new DASH client with the new ad. These delays depend entirely on application implementer and scalability of the ad servers.

In the server-driven ad insertion model the splicing delay includes the time needed for the client to request and receive a new MPD. In case remote periods are used for targeted advertisement, there is an additional delay due to XLink resolution: the client needs to request a remote period from an HTTP server which, in turn, may need to communicate with ad servers. While neither of these operations is expected to take a significant amount of time, this delay is non-negligible.

Much of this splicing delay can be compensated at the headend side. The feed itself will have the first SCTE 35 cue message announcing the upcoming ad break several seconds ahead of this break. Transcoding-related delays can also be utilized, since absolute timing of splice points does not change. Passing cues to the packager out of band as they are received at the transcoder increases the lead time and leaves the client with few extra seconds to request an MPD and resolve a remote period.

Another method of reducing delay is removing MPD update request from the picture – there are inband DASH events that allow embedding either complete MPD or an incremental difference between two MPDs. Inband MPDs should never be transmitted over an insecure channel as this allows a trivial man-in-the-middle attack.

SCALABILITY AND FAULT TOLERANCE

One of the goals of our architecture is to allow for failures of transcoders, packagers on the infrastructure side, and HTTP GET failures on the client side with no or minimum amount of disruption.

Segment request failures due to network performance can be alleviated by offering several URLs in the MPD. DASH allows specifying different URL paths for same segment, and these can point to different CDNs and/or different packagers.

Remote periods can have “default” content, thus an XLink resolution failure can be handled without disruption. Also, the packager can be configured to provide a “slate” content during a break in case a MPD update fails and the client continues to request segments according to the template of the previous period.

Packagers themselves should be able to start up fast and generate aligned bit-identical segments. If virtual segmentation is used, it is highly desirable that segmentation should start from the first segment boundary marker. This is possible if (a) template and naming convention are defined, and (b) the marker itself is carries a unique identifier that is used in the template. Virtual segmentation activities in MPEG and SCTE are considering addition of a unique counter for this purpose.

Lastly, transcoders can fail. In case some of representations are lost but service still continues, a packager can create a new period without the missing representations for the duration of the outage. Similarly a new period with the complete representation listing shall be generated when the streams become available again.

If the source is completely lost and the packager needs to switch to a different redundant transcoder, the presentation time between the first transcoder failure and the start of the first segment created by the redundant transcoder is lost. In this case a packager can insert an empty period that indicates to the client that some segments are unavailable. A simpler mitigation strategy is using a pre-encoded slate.

MONITORING AND VERIFICATION

As a rule, input and output of every workflow stage need to be validated. This means ensuring correctness the feed itself, intermediate formats and DASH content itself. A second stage is cross-validation: segments and manifests, splice time and conditioning, etc. The third stage is timing behaviour – segment and MPD generation timing, event timing, etc.

A growing amount of work and products provide client-side analytics – such as information on network state and client performance as seen by the client. These provide insights that are harder to deduce from in-network probes. Original DASH metrics and a new MPEG standardization effort, SAND (server and network assisted DASH) also address these issues

Monitoring and verification tools for traditional distribution are mature – and cover many parts of the hybrid workflow needs, such as monitoring picture quality, lip sync, closed captioning, etc. Unfortunately, many existing tools are not scalable enough – especially when we look at performance-hungry UltraHD and high frame rate workflows.

CONCLUSION

In this paper we went over the DASH implementation of ad-supported linear service and issues an architect should take into account when considering such a deployment. We further show that integration of DASH MLO with ATS linear packagers based provides a more robust, operationally simpler and fault tolerant system.

BIBLIOGRAPHY

[1] I. Sodagar, “The MPEG-DASH Standard for Multimedia Streaming Over the Internet”, IEEE Multimedia, October–December 2011, pp. 62–67

[2] T. Stockhammer, “Dynamic Adaptive Streaming over HTTP – Design Principles and Standards” In: MMSys ’11: Proceedings of the second annual ACM conference on Multimedia systems, Feb 2011, S. 133-144

[3] A. Giladi, MPEG DASH: A Brief Introduction, IEEE COMSOC MMTC E-Letter, vol. 8, no. 2, March 2013.

[4] Y Syed and A. Giladi, Hybrid Content Generation Workflows: from MPEG-2 TS to Adaptive Streaming and ISO-BMFF, In: INTX 2015

[5] Jan van der Meer, Fundamental and Evolution of MPEG-2 Systems: Paving the MPEG Road, John Wiley & Sons, West Sussex, United Kingdom, 2014

[6] ISO/IEC 23009-1:2014 Information technology -- Dynamic adaptive streaming over HTTP (DASH) -- Part 1: Media presentation description and segment formats.

[7] ITU-T Rec. H.222.0 — ISO/IEC 13818-1,Information technology -- Generic coding of moving pictures and associated audio information: Systems

[8] DASH-IF Interoperability Points, version 3.0, available at http://dashif.org/wp-content/uploads/2015/04/DASH-IF-IOP-v3.0.pdf

[9] ANSI/SCTE 35 2014, Digital Program Insertion Cueing Message for Cable

[10] SCTE DVS 1196, Adaptive Transport Stream.

[11] DVS 1202, MPEG DASH for IP-Based Cable Services, Part 1: MPD Constraints and Extensions

[12] DVS 1208, MPEG DASH for IP-Based Cable Services, Part 3: DASH/FF Profile

[13] Encoder Boundary Point Specification, http://www.cablelabs.com/wp-content/uploads/specdocs/OC-SP-EBP-I01-130118.pdf