Live Transcoding and Streaming-as-a-Service with Low Delay and High QoE

Christian Timmerer[†,‡], Daniel Weinberger[‡], Martin Smole[‡], Reinhard Grandl[‡], Christopher Mueller[‡], and Stefan Lederer[‡]
[‡]bitmovin GmbH, [†]Alpen-Adria-Universität
Klagenfurt, Austria
[‡]{*firstname.lastname*}@bitmovin.com, [†]{*firstname.lastname*}@itec.aau.at

Abstract - *Multimedia content delivery and real-time streaming over the top of the existing infrastructure is nowadays part and parcel of every media ecosystem thanks to open standards and the adoption of the Hypertext Transfer Protocol (HTTP) as its primary mean for transportation. Hardware encoder manufacturers have adopted their product lines to support the dynamic adaptive streaming over HTTP but suffer from the inflexibility to provide scalability on demand, specifically for event-based live services that are only offered for a limited period of time. The cloud computing paradigm allows for this kind of flexibility and provide the necessary elasticity in order to easily scale with the demand required for such use case scenarios. In this paper we describe how to deploy a transcoding and streaming-as-a-service platform based on open standards (i.e., mainly MPEG-DASH) utilizing standard cloud and content delivery infrastructures to enable low-delay and high-quality streaming to heterogeneous clients. We describe how to deploy it for video on demand, 24/7 live, and event-based live services.*

INTRODUCTION

Real-time entertainment services such as streaming video and audio are currently accounting for more than 70% of the Internet traffic, e.g., in North America's fixed access networks during peak periods as shown in Figure 1 [1]. Interestingly, these services are all delivered over-the-top (OTT) of the existing (networking) infrastructure using the Hypertext Transfer Protocol (HTTP) which resulted in the standardization of MPEG Dynamic Adaptive Streaming over HTTP (DASH) [2]. The MPEG-DASH standard enables smooth multimedia streaming towards heterogeneous devices and commonly assumes the usage of HTTP-URLs to identify the available segments [3].

More and more services are getting deployed adopting the MPEG-DASH standard and we see an increasing offer of various live events – 24/7 or special events for a limited time – which are solely delivered over the open Internet without any quality guarantees. Most of these services are offered for free including advertisements, which provide service providers means for monetization. In this paper we present research that led to the deployment of a live transcoding and streaming-as-a-service platform using the MPEG-DASH standard which is used for both live 24/7 and event-based temporary services. The system architecture is described and we provide details about our live transcoding and streaming-as-a-service. Additionally, we describe client support mechanisms for live low-delay streaming enabling high Quality of Experience (QoE). Finally, we provide a conclusion highlighting the major components and features introduced in this paper.

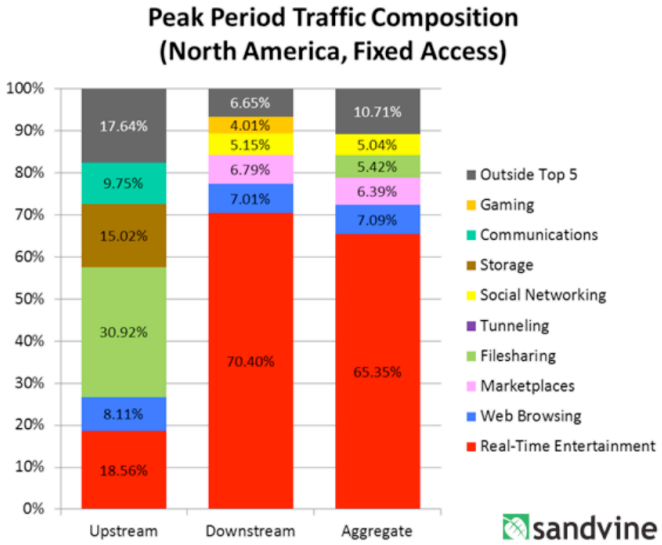

Figure 1. Peak Period Aggregation Traffic Composition - North America, Fixed Access; Real-Time Entertainment now accounts for >70% of the Internet Traffic [1].

SYSTEM ARCHITECTURE

The high-level system architecture is depicted in Figure 2. It comprises the following components (blue-rimmed modules are further detailed in this paper):

a. the actual *transcoding and streaming-as-a-service* is deployed on standard cloud infrastructure taking the live source as input and providing multiple representations (e.g., resolution, bitrate, etc.) according to the MPEG-DASH standard as output;
b. the integration within the *customer Web portal* for the actual deployment;
c. the streaming utilizing *standard delivery infrastructure* over a content distribution network (CDN); and
d. the DASH client implementation integrated within heterogeneous devices.

Figure 2: High-Level System Architecture for Live Transcoding and Streaming-as-a-Service.

The transcoding and streaming-as-a-service takes the live multimedia content as an input and transcodes it to multiple content representations in real-time on standard infrastructure-as-a-service (IaaS) cloud environments according to the requirements of the customer in terms of resolutions, bitrates, quality, etc. These requirements are expressed through an application programming interface (API) exposed to the customer. The resulting manifest describing the individual content representations and primary input for the streaming client is incorporated within the customer's Web portal offering the service to the actual clients (i.e., end users). The streaming is conducted utilizing standard CDN infrastructure. The heterogeneous clients request the multimedia segments based on the manifest received prior to the streaming and adapt themselves to the context conditions such as fluctuating network bandwidth.

Please note that our approach is explicitly targeting MPEG-DASH as the primary multimedia format representation taking into account guidelines provided by the DASH Industry Forum (DASH-IF: http://www.dashif.org). In practice, however, there is also a need to support Apple's HTTP Live Streaming (HLS) for iOS-based devices such as iPhone and iPad. This is a requirement for iOS apps submitted for distribution in their App Store. Thus, we support also HLS (http://www.dash-player.com/demo/hls/) in addition to MPEG-DASH but we hope that Apple will relax this requirement in the near future.

LIVE TRANSCODING AND STREAMING

The core of such a transcoding and streaming-as-a-service platform is the ability to take multimedia content from a live source and to transcode it in real-time – actually much faster than real-time is possible – into various content representations based on a given configuration (e.g., video profile, video quality, video resolution, audio/video bitrate). The input is typically provided using the proprietary Real-Time Messaging Protocol (RTMP) push, which is a de-facto standard used within the industry to push live content over the Internet. Other open standards such as HTTP/2 push could be a replacement but it is not yet widely available, as it has been only standardized recently [4]. Therefore, we still have to stick with RTMP push for a while.

Such a service shall support a variety of input formats in terms of video, audio, and containers as well as subtitles.

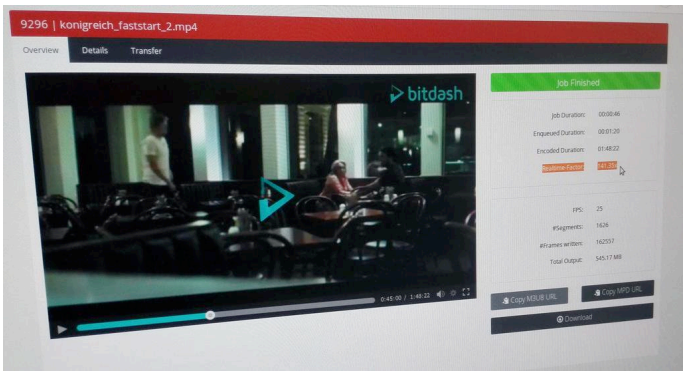

Figure 3: Screen Shot of Customer Portal showing 141x Real-Time Transcoding [Source: Sep. 2015 https://twitter.com/bitmovin/status/639505160088256512].

The transcoding mechanism utilizes the flexibility and elasticity of existing cloud infrastructure-as-a-service (IaaS) providing scalability on demand when it is needed. In particular, cloud instances are requested and utilized depending on the demand in order to satisfy real-time requirements and even beyond, i.e., transcoding to various content representations ranging from standard definition resolution to ultra high definition resolution multiple times faster than real-time. A screen shot is shown in Figure 3, which reveals the performance of being much faster than real-time and a preview of the results while the transcoding is still in progress.

In the following we will describe an example workflow in order to show how easy it is to setup a live transcoding and streaming [5]. It simply requires an arbitrary *stream name*, a stream key (required to be same as used by the ingest application/service), the timeshift buffer in seconds (if needed), the encoding profile (how many representations, bitrates, resolutions, etc.), and the output location such as Google Cloud Storage (GCS) to which all segments and the manifests are transferred. The advantage of GCS is that it offers *CDN Interconnect* for several CDNs such as Fastly, Level3, HighWinds, and CloudFlare. After starting the live stream, a RTMP URL is presented to the user which has to be added to the ingest application/service together with the stream key that is required to identify the live stream within the transcoding and streaming service.

A REST API enables easy integration into existing media workflows as well as support for multiple CDNs depending on the costumer needs.

CLIENT-SUPPORT FOR LIVE STREAMING

The MPEG-DASH standard defines the media presentation description (MPD) as well as segment formats and deliberately excludes the specification of the client behavior, i.e., the implementation of the adaptation logic, which determines the scheduling of the segment requests, is left open for competition. In the past, various implementations of the adaptation logic have been proposed both within the research community and industry deployments/products. In any case, the behavior of the adaptation logic directly impacts the Quality of Experience (QoE) which can be defined as "the degree of delight or annoyance of the user of an application or service. It results from the fulfillment of his or her expectations with respect to the utility and/or enjoyment of the application or service in the light of the user's personality and current state" [6]. For DASH-based services the main QoE influence factors can be described as initial/start-up delay, buffer underruns also known as stalls, quality switches, and media throughput [7].

I. Low-Delay Streaming

For DASH-based live services the initial or start-up delay is an important aspect for which we have developed a specific solution. The initial or start-up delay comprises the time between service/content request and start of the actual playout which typically involves processing time both at the server and client, network time for sending the MPD request and receiving first segments, and initial buffer time before the playout starts. In general, the start-up delay shall be low but it also depends on the use case. For example, the QoE of live streams or short movie clips is more sensitive to start-up delay than full-length video on demand content. Therefore, we propose a solution targeting live services using the live profile which includes a live edge hint within the MPD that allows DASH clients effectively determining the live edge. In particular, the proposed solution comprises an additional attribute to be included within the `SegmentTemplate` element in combination with the "`$Number$`" identifier for URL templates. This additional attribute is called `liveEdgeNumber` and provides the latest number of the segment which has been just written by the server upon MPD request from a client. A sequence diagram of the proposed solution is shown in Figure 4.

In contrast to conventional live-edge detection methods (i.e., calculating a starting point and searching the timeline in both directions) with `liveEdgeNumber`, the start-up delay introduced by the live-edge calculation and/or searching the timeline would decrease to zero. This mechanism can be used for all adaptive streaming systems where segments are using a "`$Number$`" identifier for URL templates. We have also deployed this solution within bitcodin demonstrating its scalability in real-world deployments.

Achieving a low end-to-end delay in live web video scenarios can be challenging, especially in context of

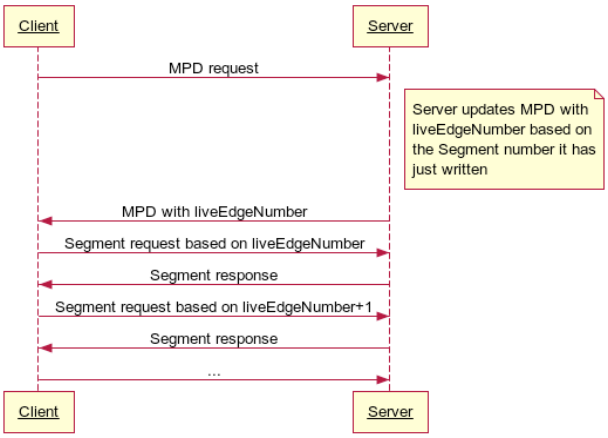

Figure 4: Sequence Diagram for liveEdgeNumber.

adaptive streaming technologies, like MPEG-DASH or HLS. There are several different components, which add additional delay to the whole end-to-end chain, like the recording device itself, the encoding instance, the player implementation and obviously the interconnection of these parts. In combination with the live transcoding and streaming service it is possible to achieve an end-to-end delay of less than 10 seconds which satisfies the requirements of a significant majority of the broadcasters as indicated in [8]. For low-delay streaming, the segment duration shall be kept small, in the order of one second, and to ensure only minor additional delay during the transmission, the livestream should be connected to a high-performance output, which can be used as origin for a CDN.

II. High-Quality Streaming

A survey on Quality of Experience of HTTP adaptive streaming [9] reveals that the "quality adaptation in video streaming and its influence on QoE is not well understood so far" and, thus, we would like to present some of the major QoE influence factors in more detail.

In previous work [7] we have presented details how to evaluate the QoE of DASH-based services describing an evaluation setup for both objective and subjective tests. The former can be done within a simple lab environment and the latter adopted a crowdsourcing approach in order to avoid cost-intensive subjective quality assessments. Seufert et al. [9] conclude that "the start-up delay does not necessarily influence the QoE but buffer underruns or stalls will definitely and also significantly impact the media experience and, thus, shall be avoided at all". Therefore, the adaptation logic should support and also be configured in a way to avoid stalls by taking into account the available bandwidth and the client's buffer state. The former can be estimated when measuring the throughput while downloading the segments and comparing it with what is provided within the MPD. The latter can be used to compensate for short-term changes in the available network.

In typical deployments player implementations try to achieve a low start-up delay by downloading the lowest quality representations first and after a while switching up to

higher quality representations (if sufficient bandwidth is available). Additionally, playback can start once enough segments are available in the client buffer (i.e., three segments are typically used) and additional low quality segments are downloaded in the background to fill up the buffer until a predefined threshold. While in some cases this is a reasonable approach, the visual quality in the beginning – depending on the number of low-quality segments buffered within the client – is usually very low which is disregarded by many users and – in the worst case – users may start to abandon the service. Therefore, player implementations should be configurable with respect to both the *start-up delay* and the *start-up quality*. The former should be a *threshold* representing the number of seconds which have to be buffered on the client before playback begins. The latter enables developers to overwrite any suggested quality level determined by the built-in adaptation logic and is referred to as *preferred start-up quality*. It can be used to create a completely new custom adaptation logic, or slightly modify the default behavior, e.g., in context of a more aggressive startup. Finally, it should be possible to specify the duration of preferred start-up quality after which the adaptation logic falls back to its default behavior. Please note that there is an intrinsic relationship between the start-up threshold and the preferred start-up delay and developers are advised to choose these values carefully.

For high quality streaming, the initial or startup delay shall be as low as possible, buffer underrun / stalls shall be zero as this would impact severely the QoE, the number of quality switches shall be also low and oscillations shall be avoided, and the media throughput at the client shall be high. Other parameters are related to the media encoding configuration including the number of representations including its bitrates, resolutions, and visual quality as well as the segment length.

CONCLUSIONS

In this paper we have presented means for the live transcoding and streaming-as-a-service with low delay and high QoE. The system architecture is presented allowing for the ingest of live content (for both professional and user-generated content) and means for much faster than real-time multi-bitrate encoding, packaging, and streaming by adopting MPEG-DASH as its primary transport format (incl. support for HLS). On the client side we proposed means for a faster start-up delay and at the same time enhanced start quality thanks to an easy-to-use configurable player implementation which – in combination with a zero-stalling adaptation logic – enables high quality streaming within heterogeneous environments.

REFERENCES

[1] Sandvine, "Global Internet Phenomena Report Africa, Middle East, and North America", *Sandvine Intelligent Broadband Networks*, December 2015.

[2] Sodogar, I., "The MPEG-DASH Standard for Multimedia Streaming over the Internet", *IEEE Multimedia,* Vol. 18, No. 4, Oct.-Dec. 2011, pp. 62-67.

[3] Timmerer, C., Mueller, C., Lederer, S., "Adaptive Media Streaming over Emerging Protocols", *Broadcast Engineering Conference (BEC)*, NAB2014, Las Vegas, NV, USA, April 2014.

[4] Belshe, M., Peon, R., Thomson, M., (eds.), "Hypertext Transfer Protocol version 2", February 2015.

[5] bitcodin blog, "MPEG-DASH and HLS Live Streaming with bitcodin", November 2015 [https://www.bitcodin.com/blog/2015/11/mpeg-dash-and-hls-live-streaming-with-bitcodin/].

[6] Le Callet, P., Möller, S., Perkis, A., (eds), "Qualinet White Paper on Definitions of Quality of Experience", *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, Lausanne, Switzerland, Version 1.2, March 2013.

[7] Timmerer, C., Weinberger, D., Mueller, C., Lederer, S. "Ultra-High-Definition-Quality of Experience with MPEG-DASH", *Broadcast Engineering Conference (BEC)*, NAB2015, Las Vegas, NV, USA, April 2015.

[8] DASH-IF, "Survey of European Broadcasters on MPEG-DASH", May 2013 [http://dashif.org/wp-content/uploads/2015/04/Survey-of-the-European-Broadcasters-on-MPEG-DASH-Whitepaper-V2.1.pdf]

[9] Seufert, M., et al., "A Survey on Quality of Experience of HTTP Adaptive Streaming", *IEEE Communications Surveys & Tutorials*, vol. 2014 (2014).

AUTHOR INFORMATION

Christian Timmerer, bitmovin GmbH and Alpen-Adria-Universität Klagenfurt, Austria, christian.timmerer@bitmovin.com.

Daniel Weinberger, bitmovin GmbH, Klagenfurt, Austria, daniel.weinberger@bitmovin.net.

Martin Smole, bitmovin GmbH, Klagenfurt, Austria, martin.smole@bitmovin.net.

Reinhard Grandl, bitmovin GmbH, Klagenfurt, Austria, reinhard.grandl@bitmovin.net.

Christopher Mueller, bitmovin GmbH, Klagenfurt, Austria, christopher.mueller@bitmovin.net.

Stefan Lederer, bitmovin GmbH, Klagenfurt, Austria, stefan.lederer@bitmovin.net.