Cloud-based Transcoding and Adaptive Video Streaming-as-a-Service

Christian Timmerer[†‡], Daniel Weinberger[‡], Martin Smole[‡], Reinhard Grandl[‡], Christopher Müller[‡], Stefan Lederer[‡]

‡bitmovin GmbH, Klagenfurt, Austria, {firsname.lastname}@bitmovin.net

†Alpen-Adria-Universität Klagenfurt, Institute of Information Technology (ITEC), Austria
christian.timmerer@itec.aau.at

1. Introduction

Real-time entertainment services such as streaming video and audio are currently accounting for more than 60% of the Internet traffic, e.g., in North America's fixed access networks during peak periods [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 segments available for the clients [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 (e.g., operas, festivals, sports) 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 *bitcodin*, a live transcoding and streaming-as-a-service platform using the MPEG-DASH standard which is – at the time of writing this paper – used for both live 24/7 and event-based temporary services and bitdash, our adaptive client framework. The system architecture is described in Section 2 and Section 3 provides details about our live transcoding and streaming-as-a-service. Quality of Experience (QoE) and client support mechanisms for live streaming are described in Section 4 and Section 5 concludes the paper. Please note that this paper has been published within IEEE ICME 2015 Industry Track [4].

2. System Architecture

The high-level system architecture bitcodin and bitdash is depicted in Figure 1. It comprises the following components (blue-rimmed modules are developed by us):

a) the actual *transcoding and streaming-as-a-service* deployed on standard cloud infrastructure (e.g., Google, Amazon, Windows, etc.) 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.

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, 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 (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, iPad, and AppleTV. This is a requirement for iOS apps submitted for distribution in their App Store. Thus, we support also HLS in addition to MPEG-DASH but we hope that Apple will relax this requirement in the near future.

3. Cloud-based Transcoding and Streaming

Figure 1: High-Level System Architecture for Live Transcoding and Streaming-as-a-Service.

The core of bitcodin 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 [5]. Therefore, we still have to stick with RTMP push for a while.

We support a variety of input formats in terms of video, audio, and containers as well as subtitles. Our 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 2, which reveals the performance of being much faster than real-time. Additionally, a preview of the results while the transcoding is still in progress is possible.

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

4. Client Implementation Framework

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

Figure 2: Screen Shot of Customer Portal showing 98.67x Real-Time Transcoding.

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**.

In order to evaluate our client bitdash we performed an objective evaluation adopting a setup similar to [7] where the bandwidth and delay between a server and client are shaped using a shell script that invokes the Unix program TC with netEM and a token bucket filter. In particular, the delay was set to 80ms and the bandwidth follows real-world bandwidth traces or a predefined trajectory comprising both abrupt and step-wise changes in the available bandwidth. The delay corresponds to what can be observed within long-distance fixed line connections or reasonable mobile networks and, thus, is representative for a broad range of application scenarios.

The test sequence is based on the available DASH dataset [8] where we adopt the Big Buck Bunny

Table 1: Comparison of Results with Commercially available Products with real-world bandwidth traces: *i)* first DASH implementation in VLC with throughput-based adaptation logic, *ii)* improvement due to HTTP persistent connections and pipelined requests, *iii)* buffer-based adaptation logic using AVC (basis for bitdash), *iv)* improvement due SVC.

Name	Average Throughput [kbps]	Switches [Number]	Stalls [s]
Microsoft Smooth Streaming	1,522	51	0
Adobe HTTP Dynamic Streaming (HDS)	1,239	97	64
Apple HTTP Live Streaming (HLS)	1,162	7	0
DASH-TPB [i]	1,045	141	0
DASH-TPB Pipelined [ii]	1,464	166	0
DASH-BB (AVC) [iii]	2,341	81	0
DASH-BB (SVC) [iv]	2,738	101	0

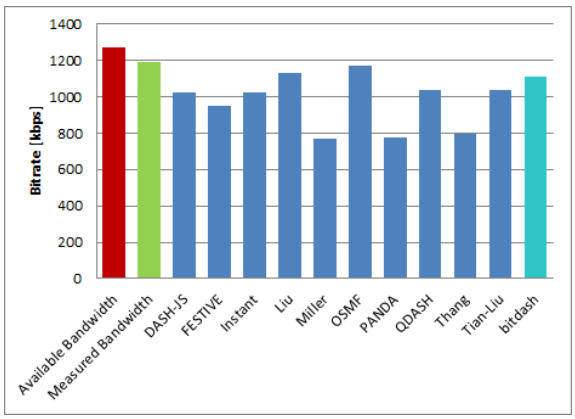

Figure 3: Average Media Throughput/Bitrate of all Adaptation Logics (higher is better).

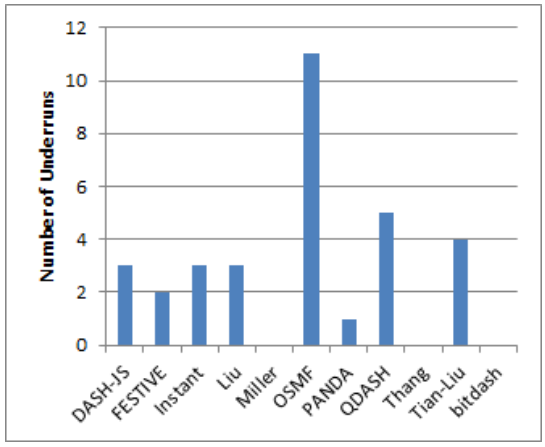

Figure 4: Number of Stalls (lower is better).

sequence providing representations with a bitrate of 100, 150, 200, 350, 500, 700, 900, 1100, 1300, 1600, 1900, 2300, 2800, 3400, 4500 kbps and resolutions ranging from 192×108 to 1920×1080. The configuration provides a good mix of resolutions and bitrates for both fixed and mobile network environments. We evaluated two versions, one with 2s segment length and one with 10s as these are the most common segment sizes currently adopted by proprietary deployments (i.e., Apple HLS uses 10s whereas others like Microsoft and Adobe use 2s).

A *first evaluation* of our DASH client implementation was based on the average throughput (in kbps), number of switches, and the duration in which the playback was stalled (in seconds). We have compared our implementation utilizing different strategies within real-world bandwidth traces to existing, proprietary solutions deployed on various platforms such as Microsoft Smooth Streaming, Adobe HTTP Dynamic Streaming (HDS), and Apple HTTP Live Streaming (HLS). The results are summarized in Table 1 and demonstrate the smoothness of the solutions offered by Microsoft and Apple but also issues with the implementation from Adobe which produced an increased number of stalls. In contrast, our first implementation of a DASH client adopts a simple throughput-based adaptation logic and provides comparable results to commercially available products, both in terms of throughput and stalls. The number of quality switches is shown to be higher but without impacting the QoE. It is known that QoE is impacted only when switching every second with a high amplitude (e.g. from high-to-low quality representation and vice-versa) [9]. Using HTTP persistent connections and pipelining increases the throughput, making it directly competitive with Microsoft Smooth Streaming. Finally, with a buffer-based adaptation we could even further increase the average throughput by reducing the number of quality switches resulting in an overall performance much better than proprietary/commercially available solutions. When using scalable video coding (SVC) a much more aggressive adaption logic can be used due the nature of layered video coding that can be exploited during streaming [9].

The *second evaluation* comprises a comparison of our buffer-based adaptation with ten different adaptation approaches proposed in the literature using the same setup as shown before but with a predefined bandwidth trajectory. The average media throughput in terms of bitrate [kbps] is shown in Figure 3. The "Available Bandwidth" on the left side of the figure shows the average bandwidth according to the predefined bandwidth trajectory used in the evaluation. The "Measured Bandwidth" by the clients is shown next to it, which is typically a bit lower than the available bandwidth due to the network overhead. The results of the different adaptation logics are shown subsequently and our implementation – on the very right side of the

figure – is among the top three implementations, namely 1. OSMF (1170.65 kbps), 2. Liu (1129.69 kbps), and 3. bitdash (1109.43 kbps). However, taking into account the average media throughput only is a fallacy when investigating the number of stalls as depicted in Figure 4. Interestingly, **among the top three, only bitdash does not produce any stalls** whereas the client with the best average media throughput produces the highest number of stalls, obviously not good for high QoE.

Finally, 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 5.

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.

5. Conclusions

In this paper we have shown a live transcoding and streaming-as-a-service – bitcodin –, which has been specifically designed for dynamic adaptive streaming over the top of the existing infrastructure using the MPEG-DASH standard (also supporting HLS for iOS apps). By using standard infrastructure, we are able to exploit the flexibility and elasticity of the cloud to provide scalability on demand for both live 24/7 services and event-based streaming for a limited time period. For the actual delivery we adopt standard content delivery networks without vendor lock-in enabling flexibility and stability for streaming services.

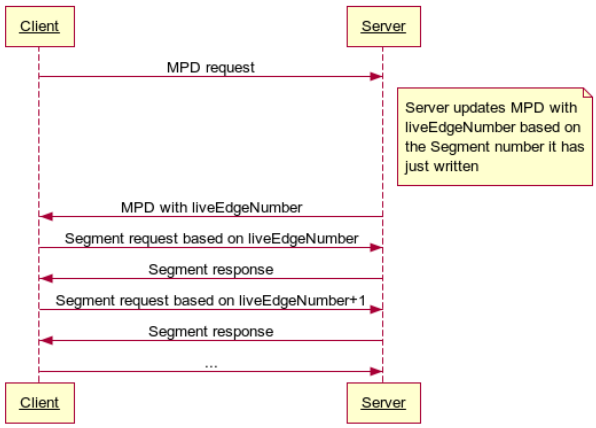

Figure 5: Sequence Diagram for liveEdgeNumber.

Finally, our streaming client offers a high average media throughput without stalling when operating in fluctuating network environments and provides instant playback for live services with a low start-up delay due to the proposed liveEdgeNumber included within the MPD and, thus, enables high Quality of Experience for the actual end user.

Acknowledgment

This work was supported in part by the EU-FP7-ICT-610370 (ICoSOLE) and Austrian FFG AdvUHD-DASH projects.

References

[1] Sandvine, "Global Internet Phenomena Report 2H 2014", Sandvine Intelligent Broadband Networks, 2014.

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

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

[4] C. Timmerer, D. Weinberger, M. Smole, R. Grandl, C. Muller, S. Lederer, "Live transcoding and streaming-as-a-service with MPEG-DASH," *in 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Jun.-Jul. 3, 2015.

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

[6] P. Le Callet, S. Möller, A. Perkis, (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, Mar. 2013.
[7] C. Mueller, S. Lederer, C. Timmerer, "An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments", *Proc. of ACM MoVid'12*, 2012.
[8] S. Lederer, C. Mueller, C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset", *Proc. ACM MMSys'12*, 2012.
[9] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys & Tutorials*, vol.17, no.1, pp. 469-492, 2015.

Christian Timmerer is an Associate Professor at Alpen-Adria-Universität Klagenfurt, Austria and his research focus is on immersive multimedia communication, streaming, adaptation, and quality of experience. He was general chair of WIAMIS 2008, QoMEX 2013, and ACM MMSys 2016 and has participated in several EC-funded projects, notably DANAE, ENTHRONE, P2P-Next, ALICANTE, SocialSensor, and the COST Action IC1003 QUALINET. Dr. Timmerer also participated in ISO/MPEG work for several years – notably, in the area of MPEG-21, MPEG-M, MPEG-V, and MPEG-DASH. He is a co-founder of bitmovin and CIA | Head of Research and Standardization. Follow him on http://www.twitter.com/timse7 and subscribe to his blog http://blog.timmerer.com.

Daniel Weinberger received a Bachelor's Degree in Computer Science from the Alpen-Adria-Universität Klagenfurt, Austria, in 2012. He is now product manager of bitdash, a Web-based adaptive streaming client, of bitmovin, Inc., a YCombinator-backed startup. His current research interests include adaptive HTTP streaming, video coding, and Web standards.

Martin Smole received his M.Sc. (Dipl.-Ing.) in 2009 from the Alpen-Adria-Universität Klagenfurt, Austria. He has gained professional experience in software development and managing software teams working in various companies including Infineon Technologies and Hewlett Packard. In 2014 he joined bitmovin where he is the product manager of bitcodin, bitmovin's cloud-based encoding service.

Reinhard Grandl received his M.Sc. (Dipl.-Ing.) from the Alpen-Adria-Universität Klagenfurt, specializing on Networked and Embedded Systems, in 2014. He joined bitmovin in 2013 as part of the player department. Now he is product manager of the bitdash player solution, focusing of adaptive streaming technologies. His current research interests include novel Internet video approaches and user-generated content streaming.

Christopher Müller is co-founder of bitmovin and CTO | Head of Technology. He received his M.Sc. (Dipl.-Ing.) from the Alpen-Adria-Universität Klagenfurt with distinction. His research interests are multimedia streaming, networking, and multimedia adaptation; he has published more than 20 papers in these areas and currently holds six U.S. patents in the area of DASH. He participated in the MPEG-DASH standardization, contributed several open source tools (VLC plugin, libdash) and participated in several EC-funded projects (ALICANTE, SocialSensor, ICoSOLE).

Stefan Lederer is co-founder of bitmovin and CEO | Head of Business. He received his his M.Sc. (Dipl.-Ing.) in Computer Science and M.Sc. (Mag.) in Business Administration from the Alpen-Adria-Universität Klagenfurt. He gained practical expertise in various companies (IBM, McKinsey&Company, Dolby, etc.) and has a strong business focus in marketing and international management. His research topics include transport of modern/rich media, multimedia adaptation, QoS/QoE and Future Media Internet architectures. He participated in several EC- funded projects (ALICANTE, SocialSensor, ICoSOLE).