Analysis and Improvement of MPEG-DASH-based Internet Live Broadcasting Services in Real-world Environments

Namgi Kim and Byoung-Dai Lee*
Division of Computer Science and Engineering, Kyonggi University
Suwon, 16227, Republic of Korea
[e-mail: {ngkim, blee}@kgu.ac.kr]
*Corresponding author: Byoung-Dai Lee

Received June 1, 2018; revised August 31, 2018; accepted December 7, 2018; published May 31, 2019

Abstract

Adaptive bitrate streaming is a crucial element in the implementation of high-quality streaming of media content over the Internet. Dynamic adaptive streaming over HTTP (MPEG-DASH) has lately emerged as the *de facto* solution for over-the-top (OTT) video streaming services. In this paper, we perform macro-level analysis on a real-world MPEG-DASH-based Internet live broadcasting service to gain insight into its behavior throughout the end-to-end service-provision chain, from broadcasters to viewers. Based on this analysis, we propose methods to improve the quality-of-experience (QoE) of MPEG-DASH-based services, particularly with regard to reducing broadcasting delays between recording and viewing of videos, as well as synchronizing client terminals.

Keywords: HTTP streaming, MPEG-DASH, MMT, live broadcasting service

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education Grants NRF-2016R1D1A1B03932696 and NRF-2017R1D1A1B04027874.

http://doi.org/10.3837/tiis.2019.05.017 **ISSN : 1976-7277**

1. Introduction

Due to improvements in video quality, increased wired and wireless network speeds, and enhancements in mobile technology in recent years, the popularity of Internet video streaming is expected to grow at a rapid pace. According to a new report by Cisco [1], by 2019, online videos will be responsible for 80% of global Internet traffic and 72% of all mobile data traffic. The report states that this drastic increase will be driven not only by the increased popularity of over-the-top (OTT) video streaming services, but also by an explosive growth in use of mobile-connected devices, such as smartphones.

The prevalence of video and mobile devices has paved the way for social platforms centered on live broadcasting. While the concept of live broadcasting has existed for years, mobile-first video platforms with user-generated content have only recently begun to have a widespread presence due to the increased ease of access to live streaming technologies and the recent rise of social media [2]. Reflecting this trend, a large number of Internet live broadcasting apps, platforms, and services, such as Periscope [3], Facebook Live [4], and YouTube Mobile Live Streaming [5], have become available over the past few years.

Adaptive bitrate streaming is essential for the implementation of high-quality media streaming over the Internet. Several proprietary technologies, such as Apple HTTP Live Streaming (HLS) [6], Microsoft Smooth Streaming [7], and Adobe's HTTP Dynamic Streaming (HDS) [8], have been developed for this purpose. However, while all of these proprietary technologies use HTTP streaming as the underlying delivery method, each one uses different protocols and data formats. Thus, to receive content from a server, a device must support the appropriate proprietary client protocol [9]. To address such interoperability issues within the industry, the Moving Picture Experts' Group (MPEG) and the 3rd Generation Partnership Project (3GPP) have developed a standard called Dynamic Adaptive Streaming over HTTP (MPEG-DASH) [10]. Since its release in 2012, MPEG-DASH has been widely deployed in commercial products ranging from connected TVs to smartphones, and has emerged as the *de facto* solution for OTT video on-demand streaming [11].

In this paper, we perform macro-level analysis on an MPEG-DASH-based Internet live broadcasting service in order to gain insight into its operation. Based on this analysis, we propose several methods to improve the quality-of-experience (QoE) of MPEG-DASH-based services, particularly through the reduction of broadcasting delays between when videos are recorded and when they are watched, as well as by enhancing playback synchronization among client terminals. The significance of our work is twofold. First, we conducted our experiments in real-world wired and wireless network environments using an MPEG-DASH-based media streaming service that is currently on the market; thus, our work is more practical than past studies in this area, most of which used synthetic MPEG-DASH-based media streaming services in controlled environments. Second, we identify the causes of two important QoE-related problems of MPEG-DASH-based media services and propose several methods to address them. These methods involve either controlling the configurable parameters of MPEG-DASH throughout the end-to-end service-provision chain, or by replacing MPEG-DASH with other suitable technologies.

The remainder of this paper is organized as follows. Section 2 introduces past research related to performance evaluations of MPEG-DASH-based media services, and Section 3 explains MPEG-DASH in brief. Section 4 describes our experimental results, and our analysis

and improvements are presented and discussed in Section 5. Section 6 contains the conclusions of this study.

2. Related Work

Since its release, significant research has been conducted on a number of issues related to MPEG-DASH, ranging from algorithm development to system and service deployment. From the perspective of performance evaluation, past research can be categorized into two different classes. The first class of studies compared MPEG-DASH against other HTTP streaming technologies, such as Apple HTTP Live Streaming and Microsoft Smooth Streaming [12]–[15], whereas the second class of studies investigated the effects of changing several configurable parameters within MPEG-DASH in order to better understand crucial factors that affect overall service performance [16]– [20]. Some recent studies related to the latter class of studies are summarized below.

Reference [16] provides basic evaluations of different segment lengths and discusses the influence of HTTP server settings on streaming performance. Their analysis was performed on a DASH dataset that consisted of full-length sequences of different genres of video, encoded at varying levels of bitrate, resolution, and quality. The authors also demonstrated how to determine the optimal segment size for a given network configuration.

In [17], the authors provide a general definition of QoE, as well as important parameters for media services that use on MPEG-DASH. In particular, they identified the main factors influencing QoE (e.g., start-up delays, media throughput, quality switching, and buffer underruns), and explained how they are related to subjective and objective evaluations of different MPEG-DASH adaptation logics.

In [18], several experiments were conducted in order to better understand "QoE unfairness," wherein concurrent users sharing the same access network receive videos of varying quality. Not only did they verify the existence of QoE unfairness, they also revealed a correlation between segment duration and users' QoE levels. According to their analysis, of two users, the one utilizing the shorter segment always received a lower QoE level.

Reference [19] shows that throughput, buffer occupancy, and variations in segment size are important factors that need to be considered in order to provide acceptable QoE through MPEG-DASH-based media services.

The authors of [20] investigated three factors that impact a user's perceived video quality: initial delay, stall (frame freezing), and bitrate (frame quality) fluctuation. Moreover, they explored multiple dimensions of each of the three factors and their resultant effects.

Although the above research has provided valuable information for better understanding the operation of MPEG-DASH-based media services, they are limited in that the relevant experiments were conducted in controlled environments. In this paper, however, we use real-world MPEG-DASH-based Internet live broadcasting services for macro-level analysis, and thus provide more practical results and insights.

3. MPEG-DASH in a Nutshell

MPEG-DASH is the first international standard for streaming video with HTTP-based adaptive bitrate. It supports interoperability between diverse devices and servers [9]. An example architecture for MPEG-DASH streaming is depicted in **Fig. 1**. The MPEG-DASH standard specifies two important data formats; the Media Presentation Description (MPD) and

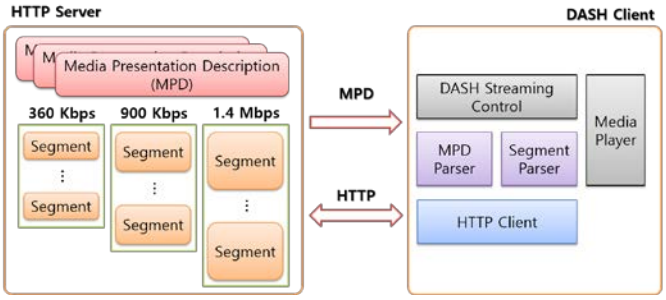

Fig. 1. An example architecture for MPEG-DASH streaming

segments. The segments contain the actual media data and the MPD provides the meta information for properly retrieving the segments. The MPD contains a manifest of available content, program timing, video resolutions and bitrates, media component locations on the network, the existence of various encoded alternatives, and other content characteristics. Thus, the DASH client has to obtain the MPD first to play the media content which is included in the segments. Based on information in the MPD, the DASH client chooses the proper encoded version of media content based on the network conditions and plays the selected version by obtaining corresponding segments. The DASH client continuously measures the network condition and selects the appropriate encoded version automatically to download and play back. Thus, it is able to seamlessly adapt to network conditions and other factors [9].

Fig. 2 shows the hierarchical structure of the MPD. The MPD consists of one or more periods, each of which has a starting time and duration to represents a program interval. It also contains one or more adaptation sets that include information concerning a set of interchangeable encoded versions using representations. The different representations within the same adaptation set describe different sorts of media content distinguished by resolution, bitrate, and other characteristics. A representation consists of at most one initialization segment and one or more media segments. The initialization segment initializes the DASH client's media decoder and the media segments deliver media data. Each segment is the largest

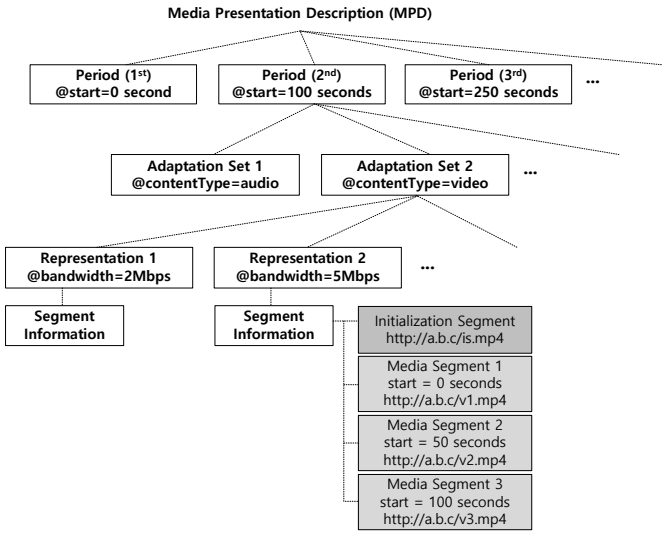

Fig. 2. Media Presentation Description (MPD) data model

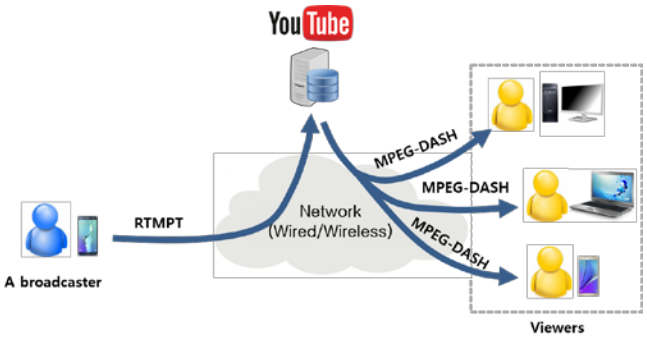

Fig. 3. The end-to-end service-provision chain for the Live Broadcast service

unit of data that can be retrieved with a single HTTP request and has a Uniform Resource Identifier (URI).

MPEG-DASH is generally implemented over HTTP/TCP. However, it is also possible to be adapted to other protocols without modifying the DASH clients. For example, in unidirectional network environments (e.g., conventional digital broadcasting systems), file delivery over unidirectional transport (FLUTE) [21] can be used as the underlying delivery protocol, instead of HTTP. In this case, segments are delivered as FLUTE objects through the file delivery table (FDT) and the HTTP-URL maps the segment URLs in the MPD [22].

4. Experimental Results

4.1 Experimental Setup and Methodology

In our experiments, we have used the Live Broadcast service, which is a camera mode available on the Samsung Galaxy series of smartphones, including the Galaxy Note 3. This mode allows users to perform live streaming sessions on YouTube using the smartphone camera. **Fig. 3** shows the end-to-end service provision chain for the Live Broadcast service. A broadcasting terminal transfers a video that is being recorded in real time to the YouTube server according to the tunneled real-time messaging protocol (RTMPT) [23], while a client terminal receives the video from the YouTube server by using the MPEG-DASH standard.

Table 1. Experimental Setups

	Model/Type	Communication Method
Broadcasting Terminal	Galaxy S6 Edge	Wi-Fi
Client Terminal	PC (Chrome Browser)	Ethernet
	Galaxy S7	Wi-Fi

As shown in **Table 1**, the Samsung Galaxy S6 Edge was used as a broadcasting terminal, and was connected through a broadband wireless network to minimize network delays during the transfer of real-time-recorded videos to the YouTube server. Since the YouTube server and the client terminal both used the MPEG-DASH standard, adaptive video streaming could be performed according to the network conditions and the performance of the client terminal. Therefore, both wired and wireless networks were used for the client terminals in order to identify the effects of network conditions on the live broadcasting service. Note that a PC connected through a wired network was used to analyze the time required for the different phases of operation (based on the MPEG-DASH standard) of the client terminal. This is because terminal performance and network conditions in the wired PC setup are superior to


```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mpeg:DASH:schema:MPD:2011" xmlns:yt="http://youtube.com/yt/2012/10/10"
xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011 DASH-MPD.xsd"
minBufferTime="PT1.500S" profiles="urn:mpeg:dash:profile:isoff-main:2011"
type="dynamic" availabilityStartTime="2016-03-08T05:45:13" timeShiftBufferDepth="PT7200.000S"
minimumUpdatePeriod="PT2.500S"
yt:earliestMediaSequence="0" yt:mpdRequestTime="2016-03-08T05:47:57.597" yt:mpdResponseTime="2016-03-08T05:47:58.798">
<Period start="PT0.000S" yt:segmentIngestTime="2016-03-08T05:45:08.257">
<SegmentList presentationTimeOffset="10997" startNumber="0" timescale="1000">
<SegmentTimeline>
<S d="1750" t="10997"/>
<S d="2499"/>
<S d="2500"/>
<S d="2499" r="1"/>
<S d="2500"/>
</SegmentTimeline>
</SegmentList>
<AdaptationSet id="0" mimeType="audio/mp4" subsegmentAlignment="true">
<Role schemeIdUri="urn:mpeg:DASH:role:2011" value="main"/>
<Representation id="140" codecs="mp4a.40.2" audioSamplingRate="48000" startWithSAP="1" bandwidth="144000">
<AudioChannelConfiguration schemeIdUri="urn:mpeg:dash:23003:3:audio_channel_configuration:2011" value="2"/>
<BaseURL>https://r4---sn-3u-bh2el.googlevideo.com/videoplayback/id/o9DV1JLVwn8.0/itag/140/source/yt_live_broadcast/requiressl/yes/play
<SegmentList>
<Initialization sourceURL="https://manifest.googlevideo.com/api/manifest/init_segment/id/o9DV1JLVwn8.0/itag/140/source/yt_live_broadca
<SegmentURL media="sq/0/clen/29389/lmt/1457415913964250/dur/1.750" mediaRange="592-"/>
<SegmentURL media="sq/1/clen/39994/lmt/1457415916067706/dur/2.499"/>
<SegmentURL media="sq/2/clen/39999/lmt/1457415918375728/dur/2.500"/>
<SegmentURL media="sq/3/clen/40012/lmt/1457415920365070/dur/2.499"/>
<SegmentURL media="sq/4/clen/40032/lmt/1457415923550094/dur/2.499"/>
<SegmentURL media="sq/5/clen/40066/lmt/1457415924903770/dur/2.500"/>
</SegmentList>
</Representation>
</AdaptationSet>
```

Fig. 4. An example MPD used in the Live Broadcast service

those in the wireless smartphone setup. This meant that the QoE in the wired PC setup more closely reflected limitations of the MPEG-DASH standard itself, rather than more extrinsic effects.

As Live Broadcast is a commercial service, the software source code for the DASH client provided in the client terminal, as well as for the HTTP server running on the YouTube servers, could not be obtained. Consequently, the macro-level analysis of the operation of the client terminal was carried out by using the network monitoring tool of the Chrome Developer Tools set [24] provided in the Google Chrome browser. When launched, the network monitoring tool recorded and reported a diverse set of information concerning each network operation on a page, including HTTP requests and response headers, a timeline of when resources were retrieved, and the total duration from the start of the request to the receipt of the final byte in the response. Therefore, it provided insights into resources requested and downloaded over the network in real time.

4.2 Experimental Results

Fig. 4 shows the details of an MPD received by the client terminal as soon as the Live Broadcast service was launched. Based on information in the MPD, the MPD update period was established as 2.5 s by the <*minimumUpdatePeriod*> element, although it was not fixed for the entire video streaming service, as it can vary across use instances. The <*SegmentTimeline*> element provided the durations of segments listed in the <*SegmentList*> element, which in turn provided information concerning the URL and the durations of all segments that could be watched once the Live Broadcast service was initiated. It should be pointed out that the segments that could be watched included all videos from the starting time of the broadcast to the given time except for those that began within the past three the segment durations. That is, if the given time was T_x and the segment duration was S_d, the video content in the section $[T_x - (S_d \times 3), T_x]$ was not included in <*SegmentList*>.

The workflow and corresponding timing information for the Live Broadcast service are shown in **Fig. 5**. When a broadcaster selected the Live Broadcast mode of the camera app, the app interacted with the YouTube server to prepare the service. Once the initialization between

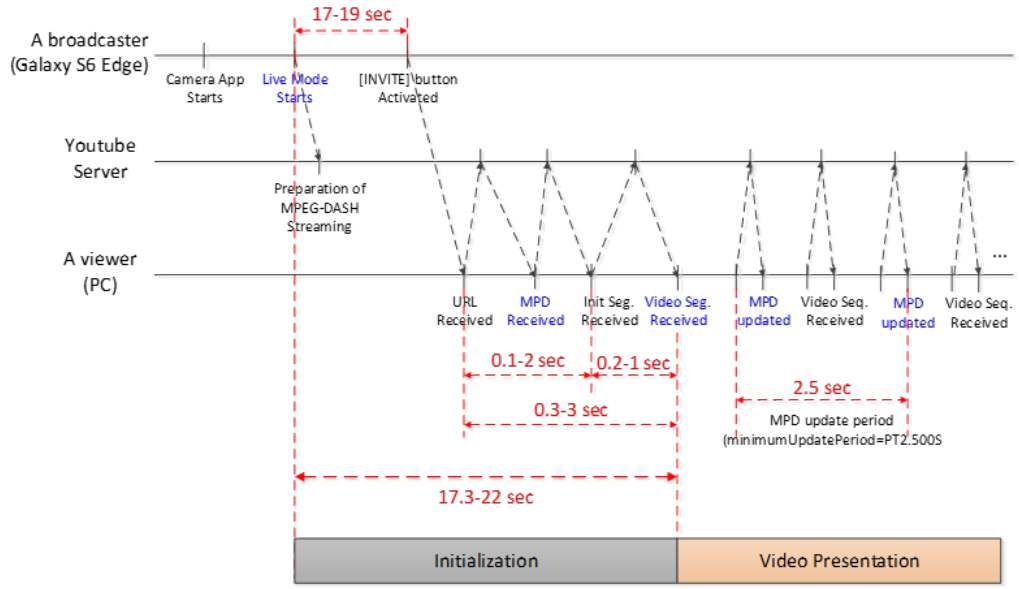

Fig. 5. Timeline of the video presentation.

the broadcaster and the YouTube server was complete, the app was ready to send the YouTube link to the live broadcast to viewers chosen from the contact list. The process required for viewers to join the live broadcasting service were the same as for the usual YouTube streaming service. Once a viewer selected the YouTube link sent by the broadcaster, their terminal began receiving the live video through MPEG-DASH.

We executed the Live Broadcast service several times and measured the average time for each phase. We found that approximately 17 to 19 s were required to initialize a YouTube server once the Live Broadcast service started – this process was not related to MPEG-DASH. After receiving the YouTube link, the viewer terminal first needed to obtain the MPD and the Initialization Segment, which took between 0.3 and 3 s. Afterward, the viewer terminal continued to retrieve video segments, where the time taken to retrieve a single video segment was between 0.2 and 1 s. Note that more time was required if the network conditions were poor. For adaptive streaming, the MPD needed to be updated every 2.5 s according to the <*minimumUpdatePeriod*> element of the previous MPD.

Although MPEG-DASH provided high-quality playback of media content by allowing seamless switching among differently coded versions of media content (based on network

a) Broadcasting delay b) Inter-client playback synchronization

Fig. 6. A demonstration of delays in MPEG-DASH-based live video broadcasting services

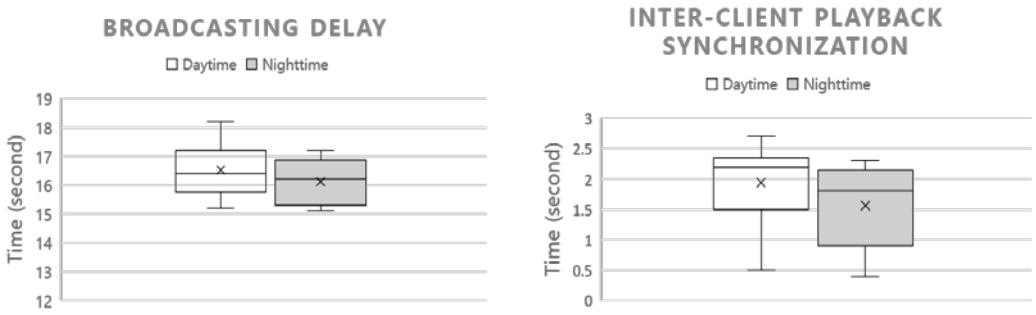

Fig. 7. Experimental results for broadcasting delays and inter-client playback synchronization.

conditions), our experimental results revealed that it suffered from two clear problems: broadcasting delays and inter-client playback synchronization. For instance, as shown in **Fig. 6(a)**, a delay of approximately 15 s occurred between the time of actual recording and viewing. **Fig. 6(b)** shows that two viewers who joined the same Live Broadcast service were presented with different sections of the broadcast. **Fig. 7** shows the boxplots of the broadcasting delays and the inter-client playback synchronization. We repeated the same experiments with the Live Broadcast service 20 times during the daytime and nighttime to measure performance variations based on network status. For both the broadcasting delay and the inter-client playback synchronization, the experimental results obtained during the nighttime slightly outperformed those obtained during the daytime in terms of average values and variances. This is very likely due to network congestion during the daytime, which would account for performance fluctuations in segment requests and transmissions.

In the following, we analyze the causes of these problems in MPEG-DASH-based live streaming services and propose methods to enhance service quality.

5. Analysis and Improvements

5.1 Analysis of Broadcasting Delays between Recording and Viewing

The delay between when videos are recorded and watched in live broadcasting services can be attributed to two factors:

① Segments available from the YouTube server

As described in the previous section, only the segments that occurred (began and ended) prior to the given time by at least three times the segment duration ($S_d \times 3$) were ready to be retrieved from the YouTube server. For example, **Fig. 8** shows that Seg#5, Seg#6, and Seg#7 were not included in the <*SegmentList*> within the MPD. Therefore, the live-streamed section of video from up to 7.5 s before the given time could not be played, which resulted in a broadcasting delay of 7.5 s between the time of recording and viewing.

② Segment first requested by the client terminal

The client terminal did not request the latest segment in <*SegmentList*> within the MPD. Instead, it requested the second-to-last segment, thus generating an additional delay of ($S_d \times 3$). For example, as shown in **Fig. 8**, only videos from Seg#0 to Seg#4 were specified in the <*SegmentList*>. Thus, Seg#2 was the first segment requested, causing a broadcasting delay of 15 s between the time of recording and time of viewing.

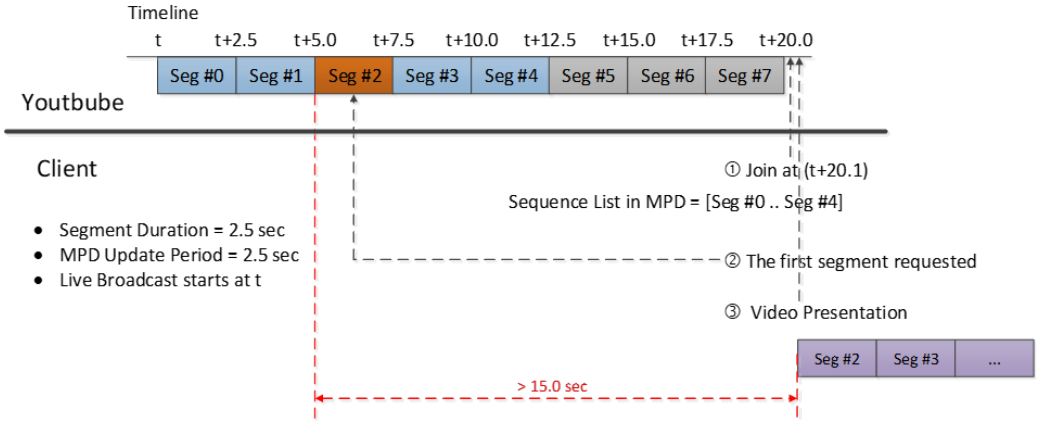

Fig. 8. Analysis of the delay between when videos were recorded and viewed (segment duration = 2.5 s)

The client terminal did not request the latest segment among the ones available due to the MPD update period. When the segment duration was shorter than the MPD update period and the segment was played to completion, the information concerning the segment that should have been continuously played may no longer have existed, thereby leading to an interruption of the broadcast. For example, **Fig. 8** shows that the MPD update period and segment duration were 2.5 s. Thus, when Seg#4 was played, the next MPD was requested after Seg#4 was already completed. Consequently, broadcasting lags occurred.

Another reason why Seg#2 was requested as opposed to Seg#3 is that the duration of each segment was not fixed to a certain value, but was varied according to the circumstances. Furthermore, the time of receiving segments could have possibly differed depending on the network conditions. Therefore, Seg#2 was requested in order to continuously populate the list of the latest segments. Therefore, the optimal first segment was determined based on the circumstances rather than on any predetermined method.

5.2 Analysis of Inter-client Playback Synchronization between Client Terminals

The playback time and duration of DASH segments were defined within the MPD (see **Fig. 4**). In particular, the playback time of segments was determined by the start attribute defined in the <*Period*> element, the "*presentationTimeOffset*" attribute in the <*SegmentList*> element, and the t attribute of the <*S*> element, which describes properties of the entire segment. The "*start*" attribute in the <*Period*> element defined the start time of the DASH Period, which was used as an anchor that determined the playback time of segments in the DASH Period.

The time at which segments in the period began playing was established to be after [*t@S* – *presentationTimeOffset@SegmentList*], relative to the *start@Period*. If the "*t*" attribute did not exist, it was calculated by adding the time at which the previous segment began playing to the duration of the given segment. As the start time of the period and the playback time of the first segment were established as 0 in the example shown in **Fig. 4**, the video was played immediately after the complete target segment was received.

MPEG-DASH determines the playback time of the segment based on relative time rather than the absolute time. Therefore, the contents of the video played could differ according to the time difference in the receiving segments. **Fig. 9** shows that if two viewers joined a live broadcast under the conditions where [Seg #0 ~ Seg #N] could be used in the YouTube server, an MPD containing an identical list of segments was received. Specifically, the same segment (Seg #N-2) was received. However, the first viewer played the target segment at time t_{x+1},

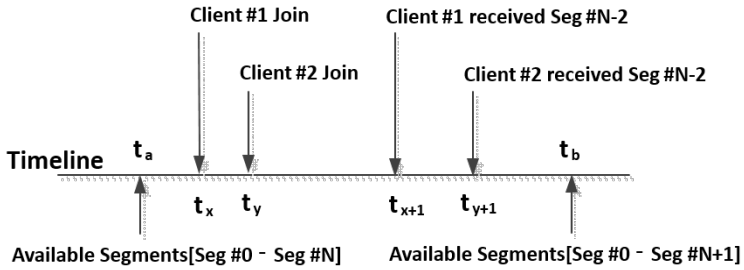

Fig. 9. Time of video playback according to the time of participation in the MPEG-DASH-based live broadcasting service

whereas the second viewer played it at time t_{y+1}. Consequently, they ended up watching different parts of the same segment at time t_{y+n}.

5.3 Enhancement Methodology for Reducing Broadcasting Delay between Videos Recorded and Watched

① Method 1: Extension of a segment range that can be used from the YouTube server

Since only the video content in the section before $[T_x — (S_{d\times3}), T_x]$ was available on the YouTube server, a delay of $(S_{d\times3})$ occurred, as discussed above. This problem can be solved by minimizing the duration of real-time segment formation in the YouTube server. However, this method was excluded from consideration in this study because it can be implemented only by YouTube, such as through an increase in resource performance and improved resource management on the YouTube server.

② Method 2: Adjustment of MPD update period

The MPD update period was shortened to solve the problem of broadcasting lags that resulted from an absence of information on the last-played segment. This reduced the broadcasting delay by $S_{d\times2}$.

When the MPD update period was established as 2.5 s as shown in the example in **Fig. 8**, an MPD update was requested five times during the playback of Seg#4 in the client terminal. Since the latest MPD containing information on Seg#5 and the following segments was received before the complete playback of Seg#4, videos could be continuously played. Moreover, the delay could be minimized to S_d (2.5 s) × 2 = 5 s by requesting Seg#4 instead of Seg#2.

The aforementioned method could generate excessive loads, as messages were frequently exchanged between the client terminal and the YouTube server due to the shortened MPD update period. Moreover, when the time taken for segment formation was longer than the segment duration in the YouTube server, information concerning new segments could not be provided until the complete playback of the target segment, thereby leading to broadcasting lags.

③ Method 3: Reduction of segment duration

When the segment duration was reduced, $S_{d\times3}$ correspondingly decreased; broadcasting delays could thusly be lowered. This method required more frequent real-time segment formation tasks in the YouTube server. It also generated a significant amount of load on the network, the client terminal, and the YouTube server, as the number of segment requests increased.

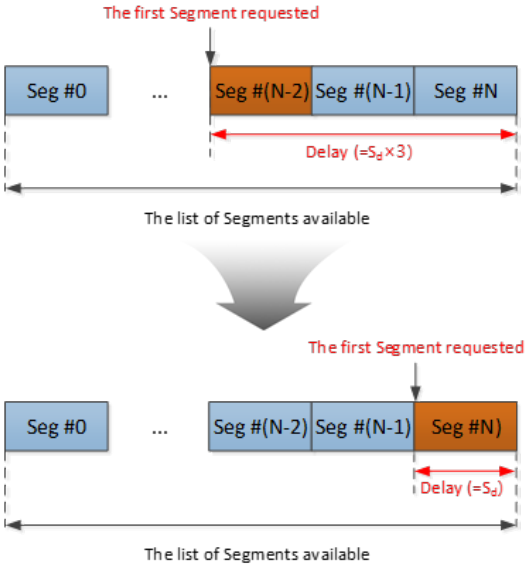

Fig. 10. Reduction in broadcasting delays through adjustment of the first requested segment.

④ Method 4: Provision of Internet live broadcasting services through MPEG Media Transport (MMT)

The PULL method of MPEG-DASH was the main cause of broadcasting delays when live broadcasting services were provided. The process of first receiving the MPD, including such signaling information as segment URLs, and then requesting new segments based on the MPD received, enabled the client terminal to dynamically request the most appropriate segment based on present network conditions, thus providing smooth media streaming services.

However, segments in the section before $S_{d\times3}$, as opposed to the most recent segments (within $S_{d\times3}$) that were already on the YouTube server, were requested to ensure sufficient time for MPD updates and prevent broadcasting lags. Consequently, the broadcasting delay increased. This problem can be solved by facilitating the initial selection of the latest segment that can be provided by the YouTube server. In this way, the broadcasting delay can be reduced to just S_d (see **Fig. 10**).

An alternative to MPEG-DASH is MPEG Media Transport (MMT) [25], which uses media processing units (MPU), which are analogous to segments in MPEG-DASH. Video and audio data, as well as signaling information, are provided through multiplexing. Thus, MMT does not need a separate process to receive signaling information (i.e. the MPD in MPEG-DASH), and is able to directly request and play the latest segment provided in the YouTube server, which reduces the broadcasting delay to S_d, as shown in the image located at the bottom of **Fig. 10**. This is because in MMT, the client terminal and the YouTube server operate via the PUSH method by using UDP/IP or Web sockets, in contrast to DASH.

Adaptive media streaming, which is considered an advantage of MPEG-DASH, can be provided through the quality-of-service (QoS) function of MMT. MPEG-DASH provides a client-based QoS function that enables the DASH client to request the most appropriate segment by evaluating network conditions. MMT provides a server-based QoS function where the MMT client can monitor network conditions and transfer relevant monitoring information

to the MMT server. The MMT server adjusts parameters related to MPU transfer based on the information received, and transfers the appropriate MPU to ensure QoS.

The pros and cons of the proposed enhancements are summarized in **Table 2**.

Table 2. Methods for Reducing the Broadcasting Delay of MPEG-DASH-based Services

Standard	Method	Effects	Potential Problems
MPEG-DASH	Reduction of MPD update period	Broadcasting delay can be minimized to S_d.	Loads on the network and the YouTube server increased due to frequent message exchanges between the client terminal and the YouTube server.
	Reduction of segment duration (S_d)	Broadcasting delay was (S_d×3). However, as Sd decreases, broadcasting delay decreased over existing S_d×3.	Loads on the client terminal and the YouTube server increased due to frequent segment requests from the client terminal.
MMT	Application of MMT	Broadcasting delay was minimized to Sd by omitting the process of receiving signaling information and updating.	Replacement of server and terminal software according to the MMT standard.
	Reduction of MPU duration (S_d)	As the size of Sd decreased, broadcasting delay was additionally reduced.	

5.4 Enhancement Methods for Solving Synchronization Problems between Client Terminals

MPEG-DASH calculates the time of video playback relative to the time the segments are received in the client terminal. Consequently, viewers who watched the same segment ended up watching different parts of the same video according to when they received the target segment. However, MMT can synchronize videos among the client terminals by describing the playback time of the video, which is included in the MPU, in terms of the coordinated universal time (UTC). Moreover, in MPEG-DASH, playback starts at the beginning of the segment, while in MMT, playback can begin anytime during the MPU, thereby allowing for excellent synchronization. For example, if the given time was Ta, and the playback time of the DASH segment and the MMT MPU was T_b (< T_a), the previous video corresponding to (T_a-T_b) was played in MPEG-DASH. Meanwhile, MMT smoothly plays the part that begins at Tb within the MPU, thereby facilitating synchronization between client terminals.

6. Conclusion

MPEG-DASH is the first adaptive bitrate HTTP-based streaming solution that is an international standard, and has become the de facto solution for OTT video streaming services. Thus, in this paper, we have analyzed a real-world MPEG-DASH-based Internet live broadcasting service at the macro level in order to better understand its operation throughout the end-to-end service-provision chain.

Demonstrated problems with MPEG-DASH-based live broadcasting services include broadcasting delays between when the videos are recorded and viewed, as well as inter-client playback synchronization issues among client terminals. According to our empirical analysis, the PULL mechanism in MPEG-DASH and the use of relative time as opposed to an absolute time are the primary causes of these problems. We have proposed several methods to address these issues.

References

[1] "Cisco VNI: Forecast and Methodology, 2017-2022," *Cisco Systems, USA*, Jun. 2016. Article (CrossRef Link).

[2] D. Mortensen, "The Live Streaming Video Report: Forecasts, Emerging Players, and Key Trends for Brands' and Publishers' Next Big Opportunity," *Business Insider*, Sep. 2016. Article (CrossRef Link).

[3] Periscope. Article (CrossRef Link).

[4] Facebook Live. Article (CrossRef Link).

[5] YouTube Mobile Live Streaming. Article (CrossRef Link).

[6] R. Pantos and E.W. May, "HTTP Live Streaming," *IETF Internet Draft*, 2017. Article (CrossRef Link).

[7] IIS Smooth Streaming Technical Overview. Article (CrossRef Link).

[8] "HTTP Dynamic Streaming Specification Version 3.0," *Adobe Systems, USA*, 2013. Article (CrossRef Link).

[9] I. Sodogar, "The MPEG-DASH standard for multimedia streaming over the Internet," *IEEE Multimedia*, Vol. 18, No. 4, pp. 62-67, Oct. 2011. Article (CrossRef Link).

[10] "Information Technology–Dynamic Adaptive Streaming over HTTP (DASH)–Part 1: Media Presentation Description and Segment Format," *ISO/IEC 23009-1*, 2014. Article (CrossRef Link).

[11] S. Zhao, Z. Li, and D. Medhi, "Low-delay MPEG DASH Streaming over the WebRTC Data Channel," in *Proc. of 2016 IEEE Int. Conf. Multimedia & Expo Workshop (ICMEW)*, pp. 1-6, Sep. 2016. Article (CrossRef Link).

[12] S. Lederer, C. Muller, and C. Timmerer, "An Evaluation of Dynamic Adaptive Streaming over HTTP in Vehicular Environments," in *Proc. of 4th Workshop on Mobile Video (MoVid)*, pp. 37-42, 2012. Article (CrossRef Link).

[13] A. Aloman, A.I. Ispas, P. Ciotimae, R. Sanchez-Iborra, and M.D. Cano, "Performance Evaluation of Video Streaming using MPEG-DASH, RTSP, and RTMP in Mobile Networks," in *Proc. of the 8th IFIP Wireless and Mobile Networking Conference (WMNC)*, pp. 5-7, Oct. 2015. Article (CrossRef Link).

[14] O. Oyman, S. Singh, "Quality-of-experience for HTTP Adaptive Streaming Services," *IEEE Communications Magazine*, Vol. 50, No. 4, pp. 20-27, Apr. 2012. Article (CrossRef Link).

[15] A. Gouta, C. Hong, D. Hong, A.-M. Kermarrec, Y. Lelouedec, "Large-scale Analysis of HTTP Adaptive Streaming in Mobile Networks," in *Proc. of 2013 IEEE 14th International Symposium on World of Wireless Mobile and Multimedia Networks (WoWMoM)*, pp. 1-10, Jun. 2013. Article (CrossRef Link).

[16] S. Lederer, C. Muller, and C. Timmerer, "Dynamic Adaptive Streaming over HTTP Dataset," in *Proc. of the 3rd Multimedia Systems Conference (MMSys12)*, pp. 89-94, Feb. 2012. Article (CrossRef Link).

[17] C. Timmer, M. Maiero, B. Rainer, and S. Petscharnig, "Quality-of-Experience of Adaptive HTTP Streaming in Real-world Environments," *IEEE CONSOC MMTC E-Letter*, Vol. 10, No. 3, pp. 6-9, May 2015. Article (CrossRef Link).

[18] A. Sideris, E. Markakis, N. Zotos, E. Pallis, and C. Skianis, "MPEG-DASH users' QoE: The Segment Duration Effect," in *Proc. of the 7th International Workshop on Quality of Multimedia Experience (QoMEX)*, pp. 262-9, Jul. 2015. Article (CrossRef Link).

[19] P. Juluri, V. Tamarapalli, and D. Medhi, "QoE Management in Dash systems Using the Segment-Aware Rate Adaptation Algorithm" in *Proc. of 2016 IEEE/IFIP Symposium on Network Operations and Management (NOMS)*, vol.10, no. 3, p. 25-29, Apr. 2016. Article (CrossRef Link).
[20] Y. Liu, S. Dey, D. Gillies, F. Ulupinar and M. Luby, "User experience Modeling for DASH Video," in *Proc. of the 20th Packet Video Workshop (PV)*, pp. 12-13, Dec. 2013. Article (CrossRef Link).
[21] T. Paila et al., "FLUTE–File Delivery over Unidirectional Transport," *IETF RFC 3926*, Nov. 2012. Article (CrossRef Link).
[22] G. K. Walker, T. Stockhammer, G. Mandyam, and Y. Wang, "ROUTE/DASH IP Streaming-based System for Delivery of Broadcast, Broadband, and Hybrid Services," *IEEE Transactions on Broadcasting*, Vol. 62, No. 1, pp. 328-337, Mar. 2016. Article (CrossRef Link).
[23] H. Parmar and M. Thornburgh, "Adobe's Real-time Messaging Protocol," *Adobe Systems, USA*, Dec. 2012. Article (CrossRef Link).
[24] "Google Chrome DevTools Overview," *Google, USA*. Article (CrossRef Link).
[25] "Information Technology–High-efficiency Coding and Media Delivery in Heterogeneous Environments–Part 1: MPEG Media Transport (MMT)," *ISO/IEC 23008-1: 2017*, 2017. Article (CrossRef Link).

Namgi Kim is an associate professor at the department of computer science, Kyonggi University, Korea. He received the B.S. degree in Computer Science from Sogang University, Korea, in 1997, and the M.S. degree and the Ph.D. degree in Computer Science from KAIST in 2000 and 2005, respectively. From 2005 to 2007, he was a research staff of the Samsung Electronics. Since 2007, he has been a faculty of the Kyonggi University. His research interests include sensor system, wireless system, and mobile communication.

Byoung-Dai Lee is an associate professor at the department of computer science, Kyonggi University, Korea. He received his B.S. and M.S. degrees in Computer Science from Yonsei University, Korea in 1996 and 1998 respectively. He received his Ph.D. degree in Computer Science and Engineering from University of Minnesota, Minneapolis, U.S.A. in 2003. Before joining the Kyonggi University, he worked at Samsung Electronics, Co., Ltd as a senior engineer from 2003 to 2010. During the period, he has participated in many commercialization projects related to mobile broadcast systems. His research interests include cloud computing, mobile multimedia platform, and mobile multimedia broadcasting. He is the corresponding author of this paper.