Industry Efforts to Unify Streaming Formats

The technologies that comprise streaming delivery can be categorized into several classes (Table 1). These technologies were proposed by a variety of companies, and many of them are now international standards. Combining individual technologies to achieve optimal streaming makes it necessary to reconcile the standards involved. Currently a great shift is beginning to take place among the "streaming formats" class of technologies.

The HTTP Live Streaming (HLS) format advocated by Apple was announced in 2009. Ever since then, this key format has played a leading role in streaming. Its distinguishing characteristic is the adoption of segmented MPEG2-TS as the container format, with small files storing video and audio data delivered in a continuous stream. The use of HTTP/1.1 as the delivery protocol also had a major impact. HLS was initially a format designed for iOS, but it later became widely used on macOS, tvOS, and Android as well.

Apple has presented a standards proposal for HLS in Internet-Draft (I-D) form to the Internet Engineering Task Force (IETF) that promotes standards for the Internet. This document is titled "draft-pantos-http-live-streaming." I-Ds are intended to be work-in-progress documents, and although some may be published as an RFC, others may never progress beyond the proposal stage. Since Apple issued the first version on May 1, 2009, this I-D has continued to be submitted under the name of Roger Pantos, and as of September 2016 it is up to version 20.

The structure of HLS is extremely simple. An example of the file containing the instructions for playback (the manifest file) is shown below (Figure 1).

The essential part of this is the resources identified by URI. In this example, the three media files "first.ts," "second.ts," and "third.ts" must be located on the media.example.com server. It doesn't matter whether these files are mounted statically or dynamically. Then, as you can see from the scheme name, it is indicated that these media files will be distributed via HTTP.

This manifest file is also placed on a Web server and passed on to the media player. The media player reads the reference information and accesses the media files listed by URI in order from top to bottom. Comment rows contain reference information, and in the case of the example shown in Figure 1, EXT-X-TARGETDURATION indicates the maximum length of the media files, and EXTINF indicates the actual length of the next media file in seconds. In other words, we can see that the media specified by this playback instruction file has a total playback duration of 21.021 seconds.

No standardization activity is being carried out for this I-D. To publish it as an RFC it would need to be discussed by an IETF working group. But this I-D has not been brought up as an ongoing issue at any working group, and has yet to be discussed. It is treated as if it were a privately published document.

In what could be interpreted as a competing standard, MPEG-DASH (Dynamic Streaming over HTTP) was standardized by the MPEG (Moving Picture Experts Group) of the International Organization for Standardization (ISO), and published as ISO/IEC 23009-1 in 2012. MPEG is a group of experts who have been active since 1988, acting as a working group for standardization tasks.

Class	Technology
Presentations	HTML5, Flash, etc.
Codecs, metadata	H.264, H.265 (HEVC), AAC, WebVTT
Containers	MPEG2-TS, MP4
Streaming formats	HLS, MPEG-DASH, CMAF
Delivery protocols	HTTP/1.1
Transport protocols Network protocols	TCP/IP

Table 1: Streaming Delivery Structure

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXTINF:9.009,
http://media.example.com/first.ts
#EXTINF:9.009,
http://media.example.com/second.ts
#EXTINF:3.003,
http://media.example.com/third.ts
#EXT-X-ENDLIST
```

Figure 1: HLS Manifest File Example

This group mainly focused on the area of standardization work for video and audio compression formats up until now, but as the streaming of video over the Internet became more popular they also branched out into drawing up streaming formats.

```
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="urn:mpeg:dash:schema:mpd:2011"
xsi:schemaLocation="urn:mpeg:dash:schema:mpd:2011
DASH-MPD.xsd" mediaPresentationDuration="PT0H1M6.1S"
minBufferTime="PT1.5S"
profiles="urn:mpeg:dash:profile:isoff-on-demand:2011"
type="static">
  <Period duration="PT0H1M6.1S" start="PT0S">
    <AdaptationSet>
      <Representation bandwidth="4000000"
codecs="avc1.4d401e" height="1080" id="1"
mimeType="video/mp4" width="1920">
        <BaseURL>video_4000.mp4</BaseURL>
        <SegmentBase indexRange="860-1023">
          <Initialization range="0-859" />
        </SegmentBase>
      </Representation>
    <Representation bandwidth="2400000"
codecs="avc1.4d401e" height="720" id="2"
mimeType="video/mp4" width="1280">
        <BaseURL>video_2400.mp4</BaseURL>
        <SegmentBase indexRange="859-1022">
          <Initialization range="0-858" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  <AdaptationSet>
    <Representation bandwidth="128000" codecs="mp4a.40.2"
id="5" mimeType="audio/mp4">
        <BaseURL>audio_128.mp4</BaseURL>
        <SegmentBase indexRange="783-946">
          <Initialization range="0-782" />
        </SegmentBase>
      </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Figure 2: MPD Example

The MPEG-DASH provisions call manifest files MPDs (Media Presentation Descriptions). Next we will show an example of an MPD (Figure 2).

You can see that it is structured using XML. The multiple Representation entries defined under AdaptationSet assume that clients will perform switching and playback dynamically. This example demonstrates that video streams of 4 Mbps and 2.4 Mbps have been prepared on the server side, enabling clients to select the optimal stream for their environment (bandwidth and CPU consumption, etc.).

The HLS, MPEG-DASH, Smooth Streaming, and HTTP Dynamic Streaming formats that are now widely used benefit greatly from adopting HTTP/1.1 as the delivery protocol. HTTP/1.1 was already quite prevalent, so it offered better scalability than using dedicated streaming protocols. The streaming formats carried over HTTP shared the idea of performing segmentation of the data and delivering small chunks from the server to the client.

However, during their design HLS, MPEG-DASH, Smooth Streaming, and HDS adopted manifest files and container formats individually, resulting in each featuring different combinations (Table 2). These circumstances caused confusion during content production and at CDN providers. When you want to support a wide range of clients, you need to create manifest files and container formats for all of them. If you are using HLS, MPEG-DASH, Smooth Streaming, and HDS, you have to create and manage four types of file. This increases the workload of production sites, and requires four times as

Name	Specification	Manifest File	Container Format	Standardization
HTTP Live Streaming	Apple	m3u8 An independent extension of the m3u standard	Segmented MPEG2-TS Latest version supports MP4	Not standardized However, it is used extensively even outside Apple
MPEG-DASH	ISO/IEC	MPD Written in XML	MP4, MPEG2-TS (MP4 is used most often)	International standard Detailed definition is carried out in various places (the MPEG Industry Forum, etc.)
Smooth Streaming	Microsoft	isml	MP4	None
HTTP Dynamic Streaming	Adobe	f4m Written in XML	f4f	None

Table 2: Common Streaming Formats

much storage to be prepared. It can also lead to decreased cache efficiency in the distribution systems of CDN providers. On top of that, the decision of which systems to implement on playback devices is a tricky issue. The adoption of Smooth Streaming and HDS has actually been dropping, with the HLS and MPEG-DASH systems chosen in most cases, but this doesn't change the fact that there are glaring inefficiencies.

Extensive structural changes have been made in the latest version 20 of HLS. It now supports MP4. Support for fragmented MP4 was added to the fragmented MPEG2-TS specified up until now. The new Packet Audio and WebVTT multimedia formats have also been added. Fragmented MP4 (fMP4) is a format also standardized by ISO/IEC that refers to a series of data sequenced into multiple files.

Support for this has created the possibility of being able to merge HLS and MPEG-DASH media libraries. If you had a single type of media library prepared using fragmented MP4, it could be distributed to multiple systems. Also from a content delivery business perspective it would provide for more efficient cache operation. This is because for both HLS and MPEG-DASH the most data-heavy part is the fragmented MP4 file group storing the video data.

Moves such as this by Apple correspond to the trend towards unifying streaming formats. In line with this, the Common Media Application Format (CMAF) has been proposed. The original draft was proposed by Apple and Microsoft, and it has been discussed by MPEG (The Moving Picture Experts Group). It is more realistic for this kind of streaming format standardization work to be carried out by MPEG rather than IETF. After all, MPEG is a community where experts in areas such as container formats and codecs gather.

The following text is given as the subtitle for the CMAF standard proposal.

Media Application Format optimized for large scale delivery of a single encrypted, adaptable multimedia presentation to a wide range of devices; compatible with a variety of adaptive streaming, broadcast, download, and storage delivery methods

It appears to comprehensively cover the technology involved, but CMAF carries over the results achieved using HLS and MPEG-DASH. To put it another way, you could say it covers all the streaming format issues that have cropped up to date.

Name	Manifest File	Containers
HTTP Live Streaming	m3u8	CMAF <CMAF internal structure> + CMAF Presentation + CMAF Selection Set (Can accommodate multiple different elements in the same content; camera footage, codecs, multi-lingual content, etc.) + CMAF Switching Set (Can accommodate multiple versions of the same content using different encoding formats) + CMAF Track + CMAF Fragments + CMAF Header
MPEG-DASH	MPD	

Table 3: Relationship Between HLS, MPEG-DASH, and CMAF

CMAF does not define manifest files, players, or delivery protocols. HLS and MPEG-DASH can be called up from the manifest files that each have.

CMAF has a hierarchical structure, and it is compatible with multiple languages and different bitrates, etc. (Table 3).

If media library creation is unified using CMAF, it will drastically reduce the work associated with creating content. Although it doesn't provide any direct benefits to users, it seems to be accepted that this is a necessary move to further popularize streaming media.

The significance of CMAF will come down to whether companies like Apple or Microsoft can set a major trend in the industry, instead of following their own paths. With this kind of framework, and the interactions between people cultivated through related discussions, a better response should be possible the next time a new challenge arises. You could say that innovation is more likely to originate from aiming to make further improvements based on what we have in place now, rather than from a unified protocol or format.

Finally, allow me to point out two challenges we face going forward.

The first is finding a way to reduce the time it takes for video viewing to start. Current live streaming usually lags behind real time by around 30 seconds or more. There are multiple reasons for this, such as the time it takes to encode and the network upload time, as well as the number of segments to buffer before initial playback. These aren't things that can be changed easily by configuring the encoder or delivery server, so they are hard to control on the stream generation side. But there is demand from people who would like to play back live event video in as close to real time as possible. This issue is widely recognized in the industry, and at some point in the future it is likely a proposal for improvement will be made.

The second challenge is offline playback. For the viewing of video on mobile devices, the download caps set by telecom carriers are a particularly large hurdle. Many people also shy away from video streaming on mobile devices. To cope with this, there is technology that downloads video to a mobile device while connected to Wi-Fi, enabling it to be viewed offline as well.

Google has added a system for offline playback to its YouTube app. You can save certain videos, and watch them offline for up to 48 hours. This function is not unlocked for all users, and is mainly enabled in countries where communications infrastructure is still developing. HLS has also put together a system for offline playback. It requires the creation of corresponding codes in an iOS app.

In both cases support is provided in some apps and systems, but when the technology begins to take off in earnest standardization will be necessary. It remains to be seen whether the industry can rise to the challenge and follow through with moves such as these.

Author:
Bunji Yamamoto
Mr. Yamamoto is a Senior Engineer in the Content Delivery and Media Business Department of the Corporate Planning Division, IIJ. He joined IIJ Media Communications in 1995 and has worked at IIJ since 2005. He is mainly involved with the development of streaming technology. Among his contributions to development of the market is the organization of the Streams-JP Mailing List, which discusses this technology.