글로벌 온라인 미디어 스트리밍 Best Practice

Part2. HTTP Streaming, Current & Next

글로벌 미디어 스트리밍 서비스 시리즈

1. Global Online Media Streaming
2. HTTP Streaming, Current & Next
3. Codec, Current & Next
4. Building HTML5 Streaming Player

글.
정영석 라임라이트 네트웍스
Advanced Service Architect

지난 회차의 "글로벌 온라인 미디어 스트리밍 Best Practice"에 이어 본 회차에서는 빠르게 발전하고 있는 OTT의 주요 기술 중 Streaming Protocol에 대해 알아보고자 한다.

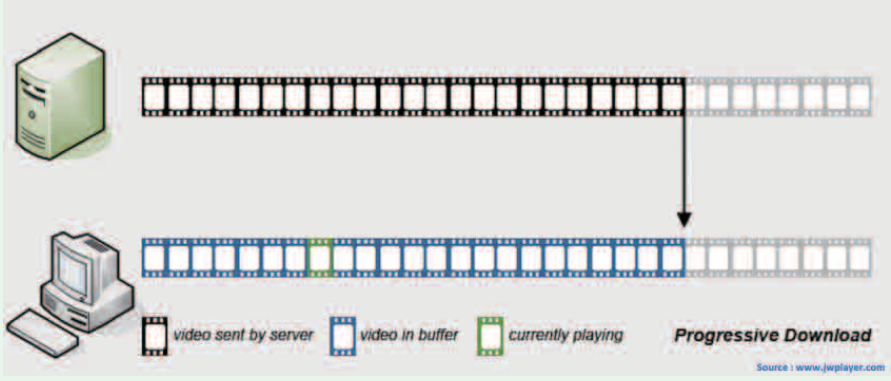
Streaming Protocol 이란?

Streaming Protocol은 인터넷을 통해 비디오 영상을 시청하는 과정에서 사용하게 되는 전송 규약을 말한다. 컴퓨터를 이용해 비디오를 시청하는 과정을 생각해보자. 인터넷 초기에는 비디오를 컴퓨터로 시청하기 위하여 인터넷을 통해 비디오 파일 전체를 다운 받아 로컬 PC에서 재생을 하였다. 영상이 완전히 다운로드 되기 전까지는 비디오를 시청할 수 없어 오랜 시간 기다려야 될 뿐 아니라 잘못된 영상을 선택하기라도 한다면 괴로움은 더욱 커지곤 했다.

Progressive Download의 등장

Apple에서는 비디오를 전부 다운로드 받아야만 영상 재생이 가능한 점을 개선하기 위해 Progressive Download라는 기술을 선보였다. 동작 원리는 점진적으로 비디오를 다운로드 받고, 다운로드 된 영상은 바로 재생이 가능하도록 하는 것이었다. 기존의 미디어 파일은 영상 메타 정보를 파일의 가장 끝부분에 배치시켜 전체 파일을 모두 받아야만 비디오 플레이어에서 재생이 가능했지만, 영상 메타 정보를 앞부분으로 옮김으로써 일부 파일만 받아도 재생이 가능하게 되었다.

아래 그림을 보면 상단의 컴퓨터가 서버이고 하단의 컴퓨터가 클라이언트이다. 화살표가 다운로드 된 부분을 표시하고 녹색 부분은 사용자가 시청하고 있는 장면이다. 네트워크 대역폭이 영상 비트레이트 보다 크다면 끊김 없이 영상을 시청할 수 있게 된다.



Progressive Download의 장점은 다음과 같이 정리할 수 있다.

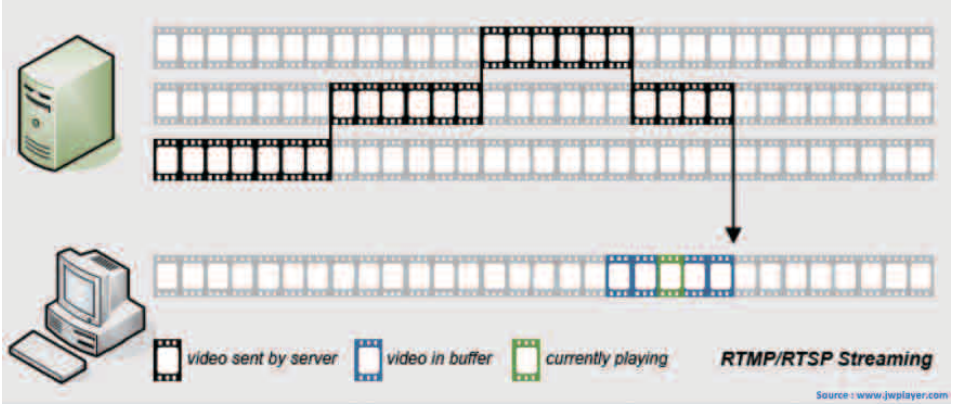
- URL을 입력한 후 거의 바로 영상 재생이 가능함
- HTTP를 사용하여 기존의 웹 서버 활용 가능 (CDN 인프라 활용 가능)

Progressive Download는 구성이 간단하여 Youtube 등 많은 서비스에서 사용되었지만, 대역폭 활용의 문제점이 있었다. 사용자가 비디오를 시청하다가 다른 비디오를 클릭하면 기존의 비디오 파일은 시청하지 않더라도 끝까지 다운로드를 받게 된다. 이는 콘텐츠 제공자 입장에서는 불필요하게 네트워크 비용을 지불하게 될 뿐 아니라 다수 사용자가 몰렸을 때 서버 한계치에 쉽게 도달하게 되는 문제가 있었다.

Streaming Protocol의 등장

Progressive Download가 대역폭의 낭비가 있을지라도 서버에 저장된 영상을 재생하기에는 충분히 좋은 기술이었다. 하지만 라이브 방송에 대한 전달은 어떻게 해야 될까? Progressive Download로 라이브 스트리밍은 불가능하다. 이때는 라이브 방송을 위한 Streaming Protocol인 RTSP(Real-Time Streaming Protocol)나 RTP(Real-time Transport Protocol) 등의 Streaming Protocol을 사용해야 했다. Streaming Protocol은 라이브뿐만 아니라 기존에 저장된 콘텐츠를 재생하는데 사용된다. 이 글을 읽고 계신 분들이라면 Real Player를 한 번쯤은 들어봤을 것이다. 초기 라이브 스트리밍 기술을 선도한 기업은 Real Player를 만든 RealNetworks이다. 이에 대응하고자 Microsoft는 MMS(Microsoft Media Services)를 내세우게 되었고, 이로써 Streaming Protocol 전쟁이 시작되었다. 2000년 중반까지는 Microsoft가 점유율을 높여갔지만, 이후 Flash 인기에 힘입어 Macromedia(Adobe가 인수)의 Streaming Protocol인 RTMP(Real-Time Messaging Protocol)에 선두 자리를 내주게 된다.

이 프로토콜들이 초기의 Streaming Protocol이며 특징은 다음과 같다.
먼저, 별도의 스트리밍 서버가 필요하다. 스트리밍 서버는 비디오 플레이어와 커넥션을 유지한 채로 비디오 재생을 한다. 커넥션을 유지함에 따라 HTTP를 이용한 전송 시 불가능했던 Adaptive Streaming이 가능하다. Adaptive Streaming은 사용자의 네트워크, CPU 등의 상태를 모니터링하여 상황에 최적화된 화질을 전송해주는 방법을 말한다. 아래 그림을 보자.



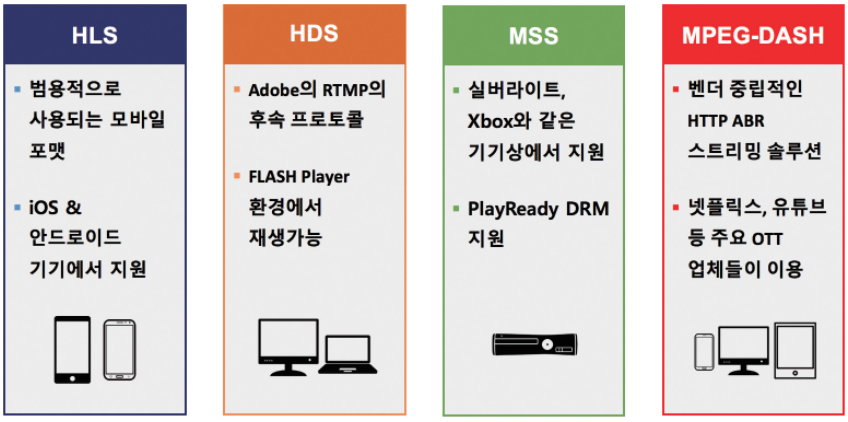
서버에 고화질/일반화질/저화질 등으로 영상을 준비해 놓고 Player에서는 Client의 네트워크나 대역폭이나 CPU 상황을 모니터링하며 상황에 따라 화질을 변경하여 사용자 환경에 최적화된 화질을 제공해 주게 된다.
또한 시청하고 있는 주변의 스트림만(위 그림의 “video in buffer”) 클라이언트의 버퍼로 저장해 놓게 된다. 즉, 도깨비 4화 시청 중 도깨비 5화로 넘어가더라도 도깨비 4화의 나머지 부분은 내려받지 않는다. 이는 네트워크 비용 절감뿐 아니라 서버의 Capacity를 효과적으로 운영하게 되는 이점이 생기게 된다.

정영석
라임라이트 네트웍스
Advanced Service
Architect

HTTP 프로토콜의 귀환

초기 Streaming Protocol 중 Flash의 인기에 힘입은 RTMP가 2010까지는 독보적으로 사용되었으나 HTML5 시대가 도래하면서 HTTP 기반 Streaming Protocol에 서서히 점유율을 빼앗기게 되었다.

Microsoft는 2008년에 MSS(Microsoft Smooth Streaming)를, Apple에서는 2009년에 HLS(HTTP Live Streaming)를, Adobe는 2010년에 HDS(HTTP Dynamic Streaming)를 각각 선보였다. 이어서 MPEG(Moving Picture Experts Group)에서도 Streaming Protocol 표준화를 위해 2011년에 MPEG-DASH를 국제 표준 규격으로 발표하였다.



왜 잘 사용하던 초기 Streaming Protocol에서 HTTP Streaming Protocol로의 전환이 이루어지고 있을까? 초기 Streaming Protocol은 HTTP 캐시를 활용할 수 없는 구조로 되어 있다. 다수의 동시 접속이 예상되면 이를 파악하여 전용 RTMP 서버를 구성해야 되는 반면 HTTP의 경우 오토스케일링이나 CDN 활용이 가능하여 비용이나 확장성 측면에서 유리하다. 즉, 초기 Streaming Protocol의 경우 고비용 저효율 아키텍처를 선택해야 되기 때문에 비용을 상대적으로 낮출 수 있는 HTTP를 선호하게 된다.

또한 W3C에서는 MSE(Media Source Extensions)라는 표준을 발표하여 MSE를 지원하는 브라우저에서는 Javascript 코드로 작성된 Player가 모두 동작할 수 있는 기반을 마련하였으며, 최신 브라우저의 경우 대부분 MSE를 지원하고 있다. 이렇듯 HTTP 기반 Streaming Protocol을 사용할 경우 재생 가능한 플랫폼이 다양해지고 있어 HTTP로의 전환이 가속화되고 있다.

HTTP Streaming Protocol의 동작 원리

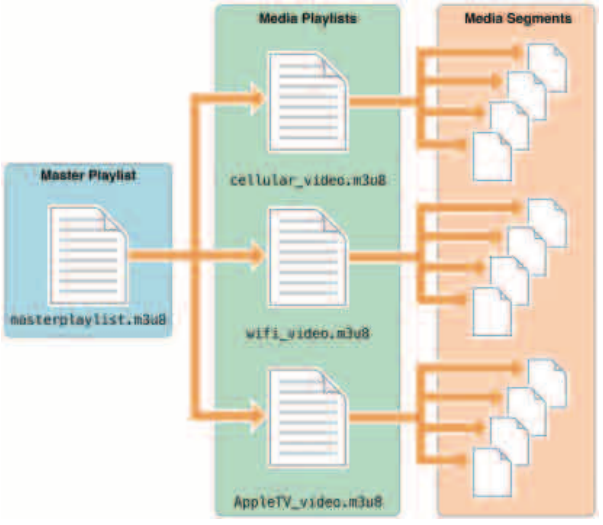
지금까지 Streaming Protocol의 태생 배경부터 기술의 흐름을 살펴보았다. 그러면 이미 대세가 된 HTTP Streaming Protocol의 동작 원리를 살펴보자. 다음 그림은 HTTP Streaming Protocol의 동작 방법을 보여준다.

서버에 고화질/일반화질/저화질(Multi-bitrate)의 영상을 준비해놓고 Client의 네트워크/CPU 상황에 따라 화질을 변경해가며 시

청을 할 수 있는 구조이다. 초기 Streaming Protocol과의 차이점은 영상 스트림을 특정 초 단위로 잘라 HTTP 웹 서버에 올려놓고 Client에서 가져갈 수 있도록 한다는 점이다. 여기서 특정 초 단위로 자르는 것을 Segmenting이나 Chunking이라 부르고 일반적으로 10초 단위로 잘라 저장한다. 이런 과정은 Streaming Engine 서버가 역할을 담당하게 된다.

HLS Workflow

Streaming Protocol의 이해를 돕기 위해 가장 널리 사용되고 있는 HLS(HTTP Live Streaming)의 처리 과정을 살펴보자. HLS는 크게 Master Playlist, Media Playlist와 Media Segment로 나뉜다.



HLS Playlist 관계도 / 출처 : developer.apple.com

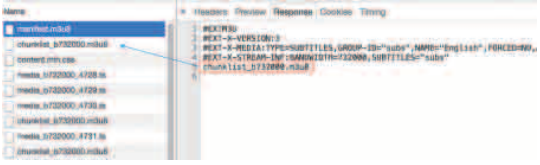
Media Segment 부분이 실제 영상을 담고 있으며, 이 세그먼트들을 순차적으로 재생하면 한편의 비디오가 완성된다. 각 세그먼트는 MPEG-2 transport stream 포맷(.ts)으로 구성되어 있으며, 보통 H.264 비디오나 AAC 오디오를 포함하고 있다. 다음은 Media Playlist 파일이다.

Media Playlist는 Media Segment의 순서 정보를 포함하여 영상이 재생될 수 있게 해주고, 앞/뒤로 점프도 가능하게 해준다. 또한, Media Playlist 파일에 광고 영상이 포함되어 있는 Segment URL을 삽입하여 광고 기능을 구현하기도 한다.

Master Playlist는 클라이언트에서 최초 서버에 접속될 때 사용되는 URL로 어떤 프로파일의(예 : 비트레이트별) 영상이 준비되어 있는지 알려준다. 클라이언트에서는 여기서 제공된 프로파일 중 재생 가능한 비디오 프로파일을 선택하여 재생하며 네트워크나 CPU 사용량에 문제가 있다고 감지되면 하위 단계의 Media Playlist로 변경하여 재생을 하게 된다. 위의 HLS 관계도에서는 3개의 비트레이트 그룹으로 나누어 단말별로 다른 프로파일을 적용하는 예를 보여 준다.

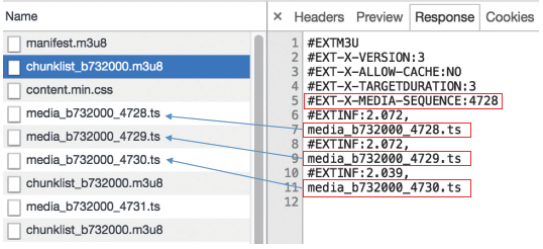
그럼 실제 HLS를 통하여 라이브 스트리밍을 재생하는 과정을 Chrome Inspector에서 살펴보자.

1) 최초 Master Playlist 요청



manifest.m3u8을 내려 받아 어떤 Media Playlist가 있는지 확인한다. 이 비디오에는 단일 비트레이트 영상만 제공되고 있음을 알 수 있다.

2) Media Playlist 요청



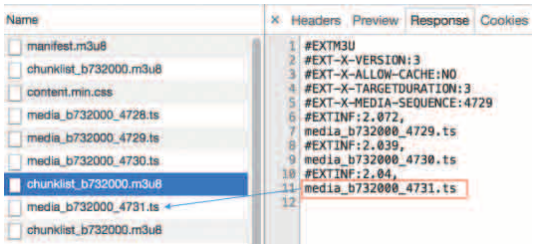
Master Playlist에 지정되어 있는 chunklist_b732000.m3u8를 내려 받아 어떤 Segment 파일을 요청해야 되는지 확인한다. EXT-X-MEDIA-SEQUENCE 번호에 해당되는 .ts부터 내려받게 된다. EXTINF 태그를 통해 제공되는 영상은 2초라는 것을 예상할 수 있다.

정영석
라임라이트 네트웍스
Advanced Service
Architect

3) Media Segment 다운로드

Media Playlist(chunklist_b732000.m3u8)에 의해 알게 된 Segment를 순서대로 내려받고 브라우저에서 영상을 재생한다. (media_b732000_4728.ts 파일을 실제로 내려받아 정보를 확인해 보면 왼쪽과 같이 H.264 비디오에 AAC 오디오를 포함하고 있는 비디오 파일로 확인된다)

4) Media Playlist 요청



media_b732000_4730.ts를 모두 받은 후 다음 Play할 Segment를 찾기 위해 Media Playlist를 다시 요청하게 한다. 이때 내려 받게 되는 응답에는 이미 내려받은 media_b732000_4729.ts와 media_b732000_4730.ts가 포함되어 있다. 이에 따라 브라우저는 media_b732000_4731.ts만 내려받게 된다. 이렇게 Media Playlist 요청 → Segment 파일 요청을 반복하며 라이브 영상을 이어가게 된다.

참고로 위의 예에서 사용된 서버는 2초 단위로 영상 세그먼트를 자르고 3개씩 내려주도록 설정해 놓았기 때문에 Media Playlist에서는 3개의 Media Segments가 보이게 된다. 이렇게 설정할 경우 인코더의 영상 입력 시점과 플레이어의 재생 시까지의 지연시간은 최소 6초 이상이 된다.

6초 (Live 시작시 3개의 영상이 준비된 후 부터 사용자에게 내려 줌)

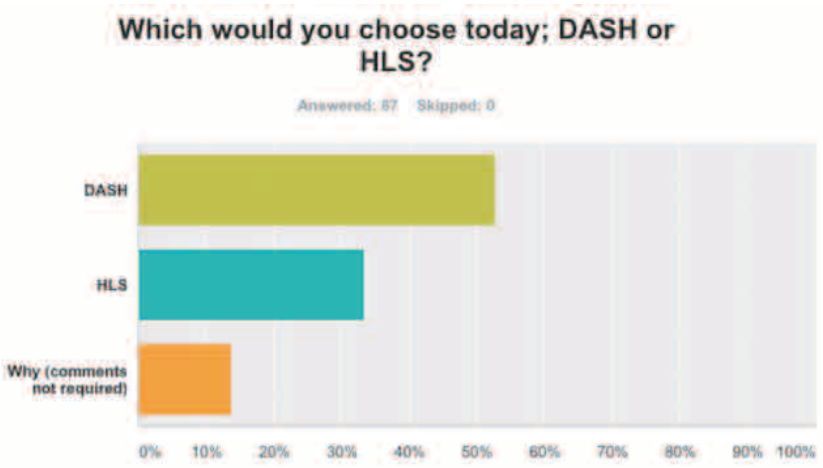
- **인코더에서 인제스트 서버에 도달하는 시간(First Mile)**
- **세그먼트 생성 시간 (Media Engine transmuxing/transcoding)**
- **사용자에게 전달되는 시간 (Last Mile)**
- **α (decoding + rendering)**

즉, HTTP Streaming Protocol을 이용한 라이브 스트리밍의 딜레이를 최소화하기 위해서는 영상 세그먼트의 길이, Playlist의 길이, 스트리밍 엔진 서버의 성능, 서버와 사용자의 네트워크 지연 시간 등을 고려하여 튜닝이 필요하다. 성공적인 스트리밍 서비스를 위한 방법에 대해서는 지난 회차의 "글로벌 온라인 미디어 스트리밍 Best Practice"를 참고하기 바란다.

Next Streaming Protocol

각 기업별로 독자적인 프로토콜을 선보임에 따라 각 기기/플랫폼별로 지원되는 HTTP Streaming Protocol이 달랐다. 이중 주도권을 먼저 잡게 된 것은 Apple의 HLS이다. Apple Device의 인기와 모바일에서의 HLS 지원에 따라 콘텐츠 제공업자는 HLS를 많이 선택하였고, 자연스럽게 HLS를 지원하는 Player와 Streaming 서버 벤더도 많아지게 되었다. 이렇게, HLS는 현재 가장 많이 사용되는 Streaming Protocol로 자리를 잡게 되었다. 그렇다면 앞으로도 HLS가 스트리밍 서비스를 할 때 가장 먼저 고려되는 프로토콜일까?

www.streaminglearningcenter.com의 조사에 따르면 이제는 HLS가 아닌 DASH를 사용하겠다고 답한 응답자가 50% 이상이다. 왜 DASH 사용을 고민할까?



출처 : www.streaminglearningcenter.com/blogs/dash-or-hls-which-is-the-best-format-today.html

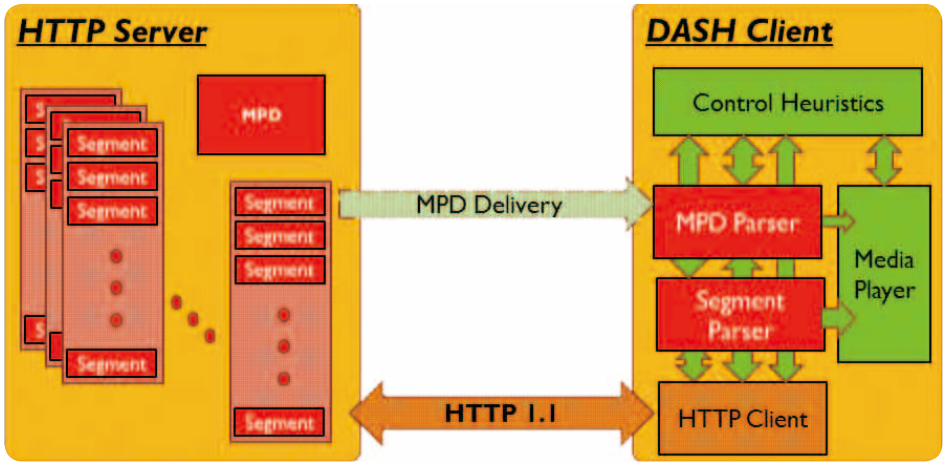
MPEG-DASH의 등장

2010년 초 HTTP Streaming Protocol은 MSS, HLS 및 HDS로 파편화되어 있었다. 프로토콜별로 지원 가능한 클라이언트가 달라 다양한 디바이스를 대상으로 서비스를 하고자 할 때는 프로토콜별 Manifest와(HLS의 Playlist에 해당) 영상 세그먼트를 준비해 놓아야 했다. 즉, 하나의 영상을 필요에 따라 다수의 프로토콜별 파일로 저장하게 되므로 스토리지 비용이 높아지게 되며 클라이언트별 호환성 여부까지 고민해야 된다. 이와 같은 파편화된 프로토콜의 문제를 개선하고자 동영상 표준화 그룹(MPEG)에서는 HTTP Streaming 표준 프로토콜 MPEG-DASH를 선보이게 되었다. (2012년에 ISO 표준으로 승격)

기본적으로 MPEG-DASH는 하나의 표준 프로토콜로 모든 디바이스에 호환이 가능하도록 목표로 하고 있으며, DRM 및 다국어 지원 등의 골치 아픈 기술적인 이슈에 대해서도 해결하고자 많은 노력을 하고 있다. 현재 Microsoft, Apple, Netflix, Qualcomm, Google, Ericsson, Samsung, Adobe 등 다양한 기업에서 MPEG-DASH 표준화에 참여하고 있어, 특정 벤더의 종속성이 없는 표준 HTTP Streaming Protocol로 자리 잡아 가고 있다.

MPEG-DASH의 범위

아래의 이미지는 DASH를 이용한 비디오 스트리밍 서비스의 HTTP 서버와 클라이언트 사이의 스트리밍 시나리오를 보여 준다.



MPEG-DASH 스펙 범위 / 출처 : White paper on MPEG-DASH Standard

정영석
라임라이트 네트웍스
Advanced Service
Architect

주요 구성은 HLS와 거의 흡사하다. MPD(Media Presentation Description)는 사용 가능한 Manifest 정보를 포함하고(ex: 고화질/일반화질/저화질 선택) Manifest는 재생할 실제 Segment URL 정보를 포함하며, Segment는 실제 영상 데이터를 포함한다. 이중 MPEG-DASH에서는 빨간색으로 표시된 MPD(Media Presentation Description)와 Segment 포맷에 대해 정의한다.

왜 MPEG-DASH인가?

MPEG-DASH의 주요 특징을 정리해보면 다음과 같다.

· **다양한 Device 지원** - HTML5를 중심으로 MPEG-DASH가 다수의 디바이스에 호환이 되도록 변화가 시작되었다. MSE(Media Source Extension)가 지원되는 브라우저라면 MPEG-DASH 재생이 가능하다. 아래 표에서 녹색으로 표시된 버전은 MSE를 지원한다는 의미이다. 보다시피 다수의 브라우저에서 MPEG-DASH 콘텐츠를 재생할 수 있는 준비가 되어 있다.

Media Source Extension지원 Browser / 출처 : caniuse.com

· **DRM 지원** - 프리미엄 콘텐츠 제공자에게는 DRM 지원 여부가 매우 중요한 항목이다. MPEG-DASH는 주요 DRM 제공사인 Adobe Primetime, Marlin, PlayReady, Widevine, Latens과 연동이 가능하며, 동시에 다수의 DRM을 지원하는 Multi-DRM 기능도 지원이 된다.

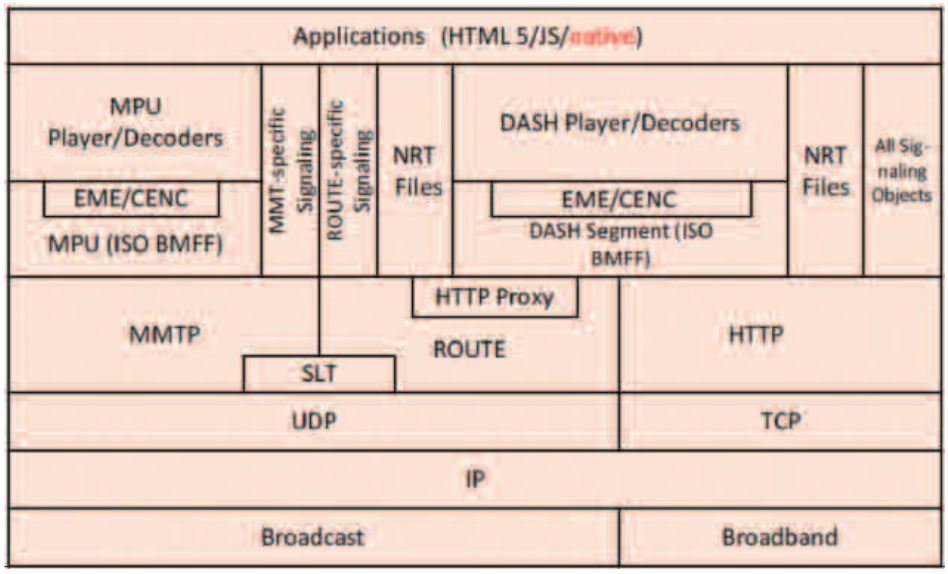
· **Video/Audio 코덱 선택의 자유로움** - MPEG-DASH는 Video나 Audio 코덱에 종속성이 없다. 새로운 코덱이 사용되더라도 클라이언트에서 지원이 된다면 동작이 가능하도록 설계되어 있다.

· **Multi-Video View - MPD(Media Presentation Description)**에 다각도의 영상 세그먼트를 명시해놓고 클라이언트에서 자연스럽게 변경해가면서 볼 수 있게 해주는 Multiple Video View를 지원한다. 예를 들어 레이싱 경기의 카메라 한 대는 운전석을 촬영하고 한 대는 트랙을 촬영한다고 할 때 이 두 영상을 하나의 화면에 보여주거나 자연스럽게 변경해가며 재생하는 것이 DASH로 가능해진다.

· **Multi-Language와 Multi-Audio 지원** - 다국어 지원을 위해서는 보통 영상 Segment에 함께 패키징을 해서 여러 개의 비디오 Segment를 준비해야 되지만 MPEG-DASH에서는 비디오/오디오/자막을 분리하여 각각 클라이언트에서 각각 내려받아 조합하는 방식으로 재생이 가능하다. 이는 서버 저장공간을 효과적으로 사용하고 콘텐츠별로 패키징하기 위한 인력과 서버 자원을 절약할 수 있게 해준다.

· **방송통신 분야의 결합** - 4K UHD 표준 기술로 잘 알려진 ATSC 3.0에서는 방송과 인터넷을 모두 지원하기 위한 하이브리드 시스템을 고려하여 표준화 작업 중이다. ATSC는 한발 앞서 있는 인터넷 분야의 비디오 스트리밍 기술을 활용하고자 TV 콘텐츠 배포 포맷을 MPEG-DASH로 선정하여 ATSC A/331 Candidate Standard를 발표하였다. 버전의 ATSC 3.0 프로토콜 스택을 봤을 때(아

래 그림 참조), 재생 방법, 영상 포맷 및 플레이어 부분은 MPEG-DASH를 따를 것으로 예상된다.



ATSC 3.0 Protocol Stack / 출처 : DASH-IF Interoperability Point for ATSC 3.0

이처럼 MPEG-DASH는 스트리밍 산업의 다양한 이슈를 해결하기 위한 방안을 제시하고 있다.

현시점에서 다수의 디바이스를 대상으로 OTT 서비스를 하고자 한다면 단연 HLS + DASH를 선택하는 것이 대부분의 선택일 것이다. MPEG-DASH를 사용 가능한 디바이스가 증가하고는 있으나 iPhone과 구버전의 브라우저를 지원하기 위해서는 어쩔 수 없이 HLS를 사용해야 된다. 이러한 조합은 앞으로도 몇 년간은 유지될 것이라는 것이 전문가들의 의견이다.

표준화된 단일 Streaming Protocol로 넘어가기 전 단일 미디어 포맷으로 표준화하기 위한 움직임도 있다. 단일 미디어 포맷으로 맞추면 HLS와 DASH에서 공통으로 사용할 수 있는 세그먼트가 만들어지는 것이다. 이는 프로토콜별 영상 패키징 비용을 줄이고 서버의 캐시 효율성을 높일 수 있다는 의미이다. 동영상 표준화 그룹(MPEG)에서는 미디어 포맷 표준화를 위해 CMAF(Common Media Application Format)를 준비 중이며 2017년 중으로 최종 버전을 발표할 예정이다. 미디어 포맷 표준화의 일환으로 Apple은 WWDC16 행사에서 fragmented-MP4를 HLS에 지원하겠다고 발표하였다. 기존에는 HLS를 위해 MPEG-2 Transport Stream으로 비디오를 패키징하고 DASH를 위해 fragmented-MP4 영상도 준비했어야 됐으나, 앞으로는 fragmented-MP4로 패키징 하면 HLS와 DASH에서 모두 사용 가능하게 된다. 이와 같은 Apple의 횡보는 스트리밍 업계에서 매우 반기고 있는 소식이며 앞으로 영상 포맷 및 스트리밍 프로토콜 단일화에 가속이 붙을 것으로 예상한다.

마치며

지금까지 Streaming Protocol의 Current & Future에 대해 살펴보았다. HTTP 스트리밍 프로토콜 표준화를 위해 많은 기업이 노력해온 결과 HLS와 MPEG-DASH가 주요 스트리밍 프로토콜로 자리 잡고 있는 추세이다. 시간이 지남에 따라 HTTP 스트리밍 업계에서는 MPEG-DASH에 더 유리한 환경을 제공해 줄 것으로 예상된다. OTT 서비스를 계획하고 있다면 MPEG-DASH, CMAF, Media Source Extension, Encrypted Media Extensions 등의 표준화 기술의 방향에 대해 살펴보면 적절한 프로토콜을 선정하는 데 도움이 될 것으로 생각한다.

다음 회차에서는 비디오 코덱의 트렌드에 대해 살펴보도록 하겠다.