Video Streaming Protocols: 
A Brief History

(Rev 1.3 – December 29, 2016)
Introduction

In this document, we will look at the evolution of protocols used for live video streaming, mostly referred as just streaming, and a complementary technology to traditional downloading. As the name implies, the technology of live streaming is typically utilized to transmit live events of various types, e.g. sports, performances, and important news in real-time. In other words, streaming is used for any type of video and audio programming that is transmitted live, usually over the Internet, as events are unfolding in real-time. A key characteristic of live streaming is that instead of playing a fixed-length media file, streaming involves transmitting a file created in real-time with neither a fixed start nor end.

Live streaming requires a device or software to capture the content live, such as a video camera and microphone, an encoder to digitize the source video, a media publisher to transcode the content into different formats and bit rates to ensure smooth delivery to a wide variety of connected devices, and a combination of Content Delivery Networks (CDN) and Internet Services Providers (ISP) to distribute and deliver the content to the end-users.

End-user devices receiving streamed content can be of any type that is enabled for the streaming protocol in use. Such devices include desktop computers and laptops, smart phones and tablets, video game consoles, and connected ("smart") TVs and IP set-top boxes (STB) such as Roku.

Live streaming has evolved over the years in response to user requirements, competitive as well as standardization efforts, while taking advantage of advances in related technologies. While it is safe to say that it has always been a complex topic, solid progress has been made. Today we find ourselves in the favorable position of being able to deliver high quality and even real HD services over the Internet, something that was hardly envisioned 10 years ago when live streaming was an exercise in end-user frustration due to video stutter and re-buffering – falling well short of a quality television experience.

OSI Model

The Open Systems Interconnection (OSI) model, which was created by the International Organization for Standardization (ISO), provides seven distinct and logical layers divided among Media Layers and Host Layers.

All streaming protocols are found in the Application layer and can therefore use the functions provided by the lower layers for support to transmit data packets. The idea is that protocols within each layer focus on providing specific functionality without the need to recreate or replicate the entire protocol stack.

A streaming protocol may use several functions in the 4th layer, i.e. the Transport Layer, to transmit packets. Some streaming protocols are more closely associated with specific Transport Layer protocols as will be shown later.
Common Streaming Protocols

This document will cover briefly the following streaming protocols:

- RTSP
- RTMP
- Adaptive Bit Rate: HLS, MSS, HDS, DASH

Broadcasting as Model for Early Streaming

Since broadcasting pre-dates streaming by some 100 years, if we consider the first radio transmission consisting of Morse Code ("wireless telegraphy") was made by Guglielmo Marconi in 1895, the approach to streaming video would initially be modeled on the former. Consequently, the video head-end was designed to control the streaming process, leaving little for the client to do apart from waiting for data packets and some buffering. Before over-the-top (OTT) video became widely popular, the norm was managed networks with controlled Quality of Services (QoS) for IPTV. In the latter case, receivers such as IP-STBs and PCs, implementing thin client architectures, will join or leave multicast streams encoded at fixed bit rates. This obviously works well for managed networks with controlled QoS. However, streaming over the Internet was a much bigger challenge due to the absence of QoS. If the bandwidth could not sustain the fixed bit rate the video would stutter, stop, and then start over, and so on, somewhat depending on the size of the local buffer, leaving the viewer frustrated.

Interestingly, what has survived from the MPEG and broadcast standards dating back more than 20 years is the MPEG-2 Transport Stream (M2TS). Today M2TS is used both for digital video broadcasting systems (cable, satellite and terrestrial), as well as in Adaptive Bit Rate (ABR) streaming protocols such as Apple's HTTP Live Streaming (HLS), and it remains an option in MPEG-DASH.

RTSP

The Real Time Streaming Protocol (RTSP) is an application-level network protocol to manage and control transfer of real-time data, such as audio and video. It is one of the traditional streaming protocols initially drafted by Real Networks and Netscape in the mid '90s. Data sources may comprise both live data feeds and files stored on media servers. The protocol is used to establish and control end-to-end media sessions by serving as a network-remote-control for time-synchronized, continuous media streams. An example of a "pull" protocol the media player issues commands such as Play, Record and Pause, enabling multimedia playback control from the media server.
RTSP is designed to control multiple end-to-end sessions, and allows for a choice of delivery channels such as UDP, multicast UDP and TCP, and delivery mechanisms based upon RTP (RFC 3550).

Before the emergence of HTTP-based ABR protocols, streaming on the Internet primarily used RTSP. While HTTP is stateless, RTSP is a so-called stateful protocol, which means that the head-end must always keep track of each client’s state, and address transmission errors in real-time while attempting to optimize the delivery.

The protocol was standardized by the Multiparty Multimedia Session Control Working Group (MMUSIC WG) of the Internet Engineering Task Force (IETF) and published as RFC 2326 in 1998. RTSP 2.0 is currently under development as a replacement of RTSP 1.0 and, generally, is not backwards compatible.

**RTMP**

The **Real-Time Messaging Protocol** (RTMP) is an application-level network protocol for delivering on-demand and live media to Adobe Flash applications. It was initially a proprietary protocol developed by Macromedia, an early leader in graphics, multimedia and web development software. The company was acquired by Adobe in 2005 and, incidentally, that year more computers worldwide had the Flash Player installed than any other Web media format, including Java, QuickTime, RealNetworks and Windows Media Player. Adobe later released an incomplete version of the Flash specification for public use (it does not include Adobe’s unique secure RTMP measures).

RTMP supports video in MP4 and FLV (Flash Video) and audio in AAC and MP3 formats. It offers several advantages compared to regular HTTP video downloads, such as robust support for live streaming, as well as dynamic streaming where the video quality adjusts on-the-fly to the available bandwidth. Media players can also seek (fast forward) to later parts in a file, especially useful for long form content.

RTMP has several direct and indirect disadvantages:

- **RTMP is only intended for use with Flash players.** Newer streaming protocols, like Apple's HTTP Live Streaming (HLS), are garnering wide industry support, as is the HTML5 standard by W3C.

- **Since RTMP streams data to the player, the bandwidth must be larger than the video bit rate.** If the bandwidth drops temporarily the stream will stutter. This can overcome by using dynamic streams that include multi-bit rate files that the player can switch between as necessary. But this is also key characteristic of modern ABR protocols and not a singular advantage of RTMP.

- **The Flash platform was also never supported by Apple for its iOS products.** The PR feud between Adobe and Apple culminated in April 2010 when Steve Jobs, the founder and then CEO of Apple, wrote his famous open letter, Thoughts on Flash. Although written more than 5 years ago, it still makes for a thought provoking read that underscores Mr. Jobs' visionary zeal.

- **While RTMP is a very robust streaming protocol and a popular CDN choice for a long time, and remains the foundation of YouTube, the Flash platform has been subjected to numerous security breaches in recent years.** Many media companies and CDNs are now transitioning from Flash/RTMP to state-of-the-art HTTP-based ABR streaming protocols. This will likely lead to the demise of RTMP over time.
HTTP-Based Adaptive Bit Rate Streaming Protocols

Adaptive bit rate (ABR) streaming is a relatively recent (~2010) technique used to stream media over IP networks. Older streaming technologies were based on protocols such as RTP with RTSP, and RTMP. Today's ABR streaming technologies are typically based on HTTP (HyperText Transfer Protocol), ensuring efficient media delivery over globally distributed HTTP networks such as the Internet. HTTP is the underlying data communication protocol used for the World Wide Web. (The other main standard that controls how the World Wide Web works is HTML, which covers how Web pages are formatted and displayed.) HTTP resides in the OSI Application Layer and will utilize a Transport Layer protocol, where Transmission Control Protocol (TCP) is used for reliable packet delivery. HTTP can also use real-time protocols such as User Datagram Protocol (UDP), which prioritizes low latency over delivery reliability.

HTTP functions as a request-response protocol in the client-server computing model. A web browser, for example, may be the client and a computer hosting media files may be the server. By design HTTP allows intermediate network elements to enable or improve client-server communications. For example, high-traffic websites frequently benefit from intermediate web caching servers that store and deliver content originating from upstream servers to reduce response time.

In the case of ABR streaming, an encoder is used to encode a single source, high bit rate video into multiple bit rate streams (i.e. quality levels), after which each of the streams is segmented into multi-second "chunks." Segment sizes vary depending on the specific implementation but are commonly between 2-10 seconds.

The key to ABR streaming is that the client device determines available bandwidth in real time, and then it chooses the video stream with the optimum quality accordingly. The client media player can switch between the different streams (bit rates) depending on available resources. This means very little buffering, fast start time and a good experience for both high and lower bandwidth connections.

The client device is informed of the available streams and their respective bit rates, together with segments of the streams, by a so-called manifest file. When the user requests the content to be played, the client device in turn requests the segments from the lowest bit rate stream. If the client determines that the available bandwidth is greater than the current segment bit rate, it will request the next higher bit rate segments. If the client realizes that the download speed for a segment is lower than the bit rate for the segment, and the available bandwidth has deteriorated, it will request a lower bit rate segment.
HTTP-based ABR technologies offer important benefits over server-driven ABR streaming:

1. The existing global HTTP delivery infrastructure of the Internet and CDNs, such as HTTP caches and servers, can be seamlessly adopted thereby drastically reducing the load on source servers.

2. Adoption of existing HTTP delivery infrastructure also reduces project risk and investments, speeding time-to-market for new operators and services.

3. Since HTTP streaming is purely client-driven, all adaptation decisions are taken by the client. This reduces the requirement of persistent connections between server and client application and enables greater scalability for a given server size and network throughput.

4. The server is not required to maintain session state information on each client, thereby also increasing scalability.

5. Since the streaming technology is built on top of HTTP, contrary to the RTP-based ABR approach, the packets easily traverse potential obstacles such as firewalls and NAT devices.

Adaptive bit rate streaming was introduced by Move Networks and further developed by Adobe Systems, Apple and Microsoft. In September 2010, Move Networks was awarded a patent for their technology innovation, and in January 2011 the company was acquired by EchoStar Corp.

The most common HTTP-based ABR streaming protocols are:

- Apple's HTTP Live Streaming (HLS)
- Microsoft HTTP Smooth Streaming (MSS),
- Adobe HTTP Dynamic Streaming (HDS)

Each of these protocols exhibit pro's and con's but it would take too long to describe them in this brief document. Suffice to say that both Microsoft and Adobe (and, to a lesser degree, Apple) are supporting the evolution to a standards-based ABR protocol, MPEG-DASH.

Criticism has been levied against HTTP-based ABR technologies for being operationally more complex than traditional streaming technologies. Encoding to multiple bit rates impacts storage requirements and CDN footprint, although significant strides have been made through new techniques such as Just-in-Time (JIT) packaging. However, these potential issues are outweighed in practice by the very favorable economics and scalability of HTTP delivery. Whereas non-HTTP ABR solutions require massive deployment of, and investments in, specialized streaming delivery infrastructure, HTTP-based ABR streaming can simply leverage the same HTTP web servers already used to deliver all other content over the Internet with the added benefit that it traverses firewalls and NAT devices.

**Standardizing on MPEG-DASH**

Dynamic Adaptive Streaming over HTTP (DASH) is the first HTTP-based ABR streaming protocol that has become an international standard, developed under the auspices of MPEG. It grew out of a grassroots initiative, the DASH Promoters Group, and was formally incorporated in September 2012 as the DASH Industry Forum (DASH-IF), now supported by 90+ member companies. The MPEG-DASH standard was first published in 2012, superseded by [MPEG-DASH 2nd Edition Specification (ISO/IEC 23009-1:2014)](https://www.iso.org/obp/topic_25560.html) in 2014.
MPEG-DASH is a technology that in principle unifies key characteristics from Adobe Systems' HDS, Apple's HLS, Microsoft's Smooth Streaming, and with many more enhancements as a result of the standards development work over the past five years. Standardizing an ABR solution is intended to inspire confidence to industry participants that the solution can be adopted for universal deployment while resolving the fragmentation stemming from vendor-centric solutions.

The MPEG-DASH standard is attractive to many companies because of some key market benefits that it brings per the DASH-IF:

- **Independent stable international standard** – not owned by any single company, DASH is a finalized specification and not a moving target.
- **Multi-video and audio tracks** – Deliver the complexity of a DVD or Blu-ray experience, with multiple synchronized video and audio options.
- **Mix of multiplexed and non-multiplexed video and audio tracks** – Provide for dynamic bandwidth adaptation, support for multiple audio options such as language selection and surround sound, bandwidth efficiency (sending only the requested tracks) and reduced production, storage, maintenance and delivery costs.
- **Common encryption** – One-time encryption and packaging of content allowing simultaneous use of multiple DRM technologies.
- **Templated manifests** – Compact manifest for fast start-up, as well as avoiding manifest download with every segment.
- **Non-segmented origin files** – Files can optionally be stored contiguously on the server and the segments accessed via byte-range requests.
- **Efficient ad insertion** – Server-based and client-based targeted ad-insertion through the use of periods.
- **Support for multiple CDNs/caches with the same manifest** – Provide flexibility to define multiple base URL in the manifest, thus improving scalability and fault tolerance
- **Accessibility, rating and other content descriptions** – Possibility of signaling content descriptions such as accessibility, rating, audio channel configuration in the manifest.
- **Industry convergence for streaming delivery** – Avoid having to provide multiple streaming solutions, each of which requires a separate ad insertion flow, content protection scheme, and a different closed captioning format.
- **Vibrant ecosystem** – Already a strong community of encoders, content packagers, delivery platforms, and player builders has been established, providing a wide range of DASH solutions.
- **Ease of integration** – DASH will work on any HTTP server and most mainstream media servers, meaning service providers and broadcast operators are not required to buy into new vendor-specific ecosystems or invest in specialized DASH-specific infrastructure.

To read about the latest MPEG-DASH standards development, read the Streaming Media magazine article *The State of MPEG-DASH 2015.*
Conclusion

While both Microsoft and Adobe offer full DASH support, together with many other technology vendors of encoders, streamers and content security/DRM, the absence of Apple backing means that DASH may not reach the universal success that many had hoped for in the early stages of the standardization. Nevertheless, the DASH standards efforts are continuing with undiminished enthusiasm by the backers.

The popularity of RTMP/Flash is assumed to be declining due to the security issues, and influenced by Adobe backing DASH as well. There is also increasing support for the HTML5 industry standard. However, while Flash is non-existent for mobile devices, Flash video playback continues to be a mainstay for desktop browsers, especially in combination with Adobe’s HTTP Dynamic Streaming. It is therefore likely that projections of Flash’s demise will continue to be unrealized until such time that a real multi-platform industry standard takes over.

Furthermore, while Apple supports HTML5 it is likely to focus on the HLS protocol for the foreseeable future. The HLS numbers for mobile video speak for themselves: Per an IDC report, Android OS and iOS together command more than 95% market share among mobile devices. Android, which now supports HLS natively, will ship on almost 1.5 billion devices in 2015 per a Gartner report. So even though the Android market is highly fragmented in terms of OS versions and devices types (more than 18,000 according to the third annual “Android fragmentation” report from the British company OpenSignal), it nevertheless means that the HLS protocol is supported across 95% of the mobile market.

While HTML5 is supported by all major desktop browsers, it is no short term mobile video contender due to fragmentation across some 15 mobile browsers supporting different levels of HTML5. An added complication is that video codecs are not yet supported in a consistent manner by HTML5 browsers for a variety of reasons, some of which are IPR related and unlikely to be easily overcome.

Apple, on the other hand, has recently announced support for fMP4 (fragmented MP4) for HLS, which is another important step towards the common goal of ABR standardization.

This may all be further brought together by the emerging Common Media Application Format (CMAF) and Common Encryption (CENC) standard although there is more work to do before a single ABR format is agreed and adopted.

Obscuring the picture further is the need for digital rights management for pay-TV services. However, DRM issues are beyond the scope of this document.

Therefore, maintaining compatibility with multiple streaming protocols will still be required for years to come. Focusing on HLS will suffice for the clear majority mobile devices while Flash will remain relevant for desktop browsers, at least until DASH gains stronger momentum. Only time will tell whether there will eventually be only one dominant live streaming protocol.