Project 3: Adobe HTTP Dynamic Streaming

Student: Vernier Marco

Student-Id: 1061001

Date: February 2011

Introduction

HTTP Dyamic Streaming is a new distribution protocol used for on-demand streaming and for live streaming. This new method of distribution allow, by the use of normal HTTP connection the on-demand sreaming and live streaming with adaptive bitstream of MP4 standard contents.
The main advantages of HTTP Dynamic Streaming, unlike the RTMP protocol, are the low latency, a faster startup, a dynamic buffer, an encryption of the stream. It also provides a better use of the storage infrastructure in the existing cache (ISP, home network etc.) and the tools to integrate the preparation of the contents in the existing encode streams.
HTTP Dynamic Streaming allow also a robust and flexible content protection for both live and on-demand content, powered by Adobe Access 2 software.

Media presentation structured and stored

HTTP Dynamic Streaming extends the F4V format using an additional standard-based on MP4 fragment format (extension .f4f). This type of format allows the use of HTTP “gets” requests to take and cache smaller portions of the media.

A F4F file is composed by these elements:

Fragments

Due to streaming issue, Media data is divided into small fragments. These fragments contain both metadata (moof) that are encapsulated in a “moov box” and media data (mdat). Every fragment starts with a random access point (for example a key frame) and can contain multiple seekable random access points.

Multiplexed hint track

To provide an efficient view experience in Flash, F4F use also Multiplexed hint tracks stored in the media as hint tracks; the multiplexed hint tracks enable easy syncronization of audio and video data at the client.

Segments

Every media presentation is composed by one or more segments. These segments are in turn composed by a collection of fragments (containing one multiplexed hint track) saved as a file or a resource. Fragments and segments are uniquely addressable by URL.

Bootstrap information

The bootstrap information are a set of information that are used to start consuming of the media data. This information, optimized for the quickest start, is located both in the media file F4F and the manifest file F4M (manifest file is a XML-based file containing different types of information about the media).

More detailed, the bootstrap contain information about fragments and segments, (stored in a fastest access run table) information about the list of web server where the data is stored and optional metadata.

The run table, where are stored the fragments and segments, contains elements that are typically expected have the same value; in this case a single record in the fragment run table represent a series of fragments with the same duration, although can be support fragment with different duration.

About the information included in the bootsrap, when F4F files are used with F4M file, the information like web server location or metadata can be stored in two different ways: could be included in the F4V media (in the "BootstrapInfo box) or saved in an external bootstrap file.

Other information:

- URL deviation scheme: are scheme used to requesting fragments within a file and is based in operations of parses of the URL by the HTTP Origin Module that return the bytes corresponding to that fragment for a specific F4F file.
- Random access point: are syncronization point in a media file used to played back a media.

Flash Media Manifest file

The manifest file is a XML-based open file that represent a single media asset and provide at the client all the necessary information to initiate the playback. Into the manifest file are contained information about the available bitrate, the video fragment file, the bootstrap information, the DRM license server location(s), the file location(s) and other metadata information for a single piece of media.

In the figure 1 you can see an example of a manifest file:

```
<?xml version="1.0" encoding="utf-8"?>
        <manifest xmlns="http://ns.adobe.com/f4m/1.0">
        <id>myvideo</id>
        <duration>253</duration>
        <mimeType>video/x-flv</mimeType>
        <streamType>recorded</streamType>
        <baseURL>http://example.com"</baseURL>
        <drmMetadata url="http://mydrmserver.com/mydrmmetadata"/>
        <bootstrapInfo profile="named" url="/mybootstrapinfo"/>
        <media url="/myvideo/medium" bitrate="908" width="800" height="600"/>
</manifest>
```

Fig.1

Another important characteristic of the manifest file, as you can see in the example below, is that supports multibitrate delivery, with each bitrate file added as a separate media node.
Files of different bitrate can share DRM and HTTP bootstrap information, or this information can be specified separately for each individual bitrate file.
Another important aspect of the manifest file is that it is customizable adding the information you want.

```
<?xml version="1.0" encoding="utf-8"?>
        <manifest xmlns="http://ns.adobe.com/f4m/1.0">
        <id>myvideo</id>
        <duration>253</duration>
        <mimeType>video/x-flv</mimeType>
        <streamType>recorded</streamType>
        <baseURL>http://example.com"</baseURL>
        <drmMetadata url="http://mydrmserver.com/mydrmmetadata"/>
        <bootstrapInfo profile="named" url="/mybootstrapinfo"/>
        <media url="/myvideo/low" bitrate="408" width="640" height="480"/>
        <media url="/myvideo/medium" bitrate="908" width="800" height="600"/>
        <media url="/myvideo/high" bitrate="1708" width="1920" height="1080"/>
</manifest>
```

Fig. 2

Manifest file workflow: beahviour of the client

The client send a normal HTTP request to the server for the manifest file that contain all the relevant information to initiate the playback. The client parses the manifest file and uses the bootstrap information contained in the F4M file and translate the desidered time code to a pair segment/number – fragment/number. The client then constructs a URL based on this number pair to request a specific fragment from a specific F4F file. This is passed to the server's HTTP Origin

Module, which references the F4X index file to locate the specific fragment requested within the F4F file. This fragment is then sent to the client for playback.
To allow this type of operation could be used the Actionscript framework OSMF (Open Source Media Framework). This is a robust and flexible Actionscript framework that allow developers to assemble components to create high-quality, full-featured video playback experience.
OSMF provides a robust open source code to create custom player that support a variety of media types as JPG, MP3, FLV, F4V and F4F.

Adaptive streaming

Adobe HTTP Dynamic Streaming provides tools to integrates adaptive streaming into encoding workflow. In this case the manifest file include information about different resources that can be provided at different bitrate. To allow this type of operation is required a key frame alignment to control the regular transition among different bitrate. The player is constantly monitoring of bandwidth changing and playback performance. For example if the bandwitdth or playback performance decrease, the player can request to switch to a lower bitrate (or higher if the bandwith is available).

Major difference to the 3GPP-DASH standard

The main differences among Adobe HTTP Dynamic Streaming and 3GPP-DASH standard can be briefly summarized in the following table:

Features	3GPP-DASH standard	Adobe HTTP Dynamic S.
On demand and live	YES	YES
Live DVR	YES	YES
Delivery protocol	HTTP	HTTP
Scalability via HTTP Edge Caches	YES	YES
Origin server	HTTP	HTTP
Stateless Server Connection	YES	YES
Media container	3GP/MP4	F4F(MP4)
DRM Support for live & VOD	OMA DRM	FLASH ACCESS DRM
Supported video codecs	AGNOSTIC	H.264/AAC
Default Segment Duration	FLEXIBLE	FLEXIBLE
End to end latency	FLEXIBLE	<30s
File type on server	FRAGMENTED	FRAGMENTED

References

Adobe Flash Platform Technical White Paper
http://www.adobe.com/products/httpdynamicstreaming/pdfs/httpdynamicstreaming_wp_ue.pdf

Open Source Media Framework
http://www.opensourcemediaframework.com/

Dynamic Adaptive Streaming over HTTP – Standards and Design Principles | Thomas Stockhammer
http://www.w3.org/2010/11/web-and-tv/papers/webtv2_submission_64.pdf

HTTP Dynamic Streaming
http://www.adobe.com/it/products/httpdynamicstreaming/