Video Adaptativo Sobre HTTP

Pablo Flores Guridi, pablof@fing.edu.uy

Curso Tecnología de Servicios Audiovisuales
Instituto de Ingeniería Eléctrica
Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

16 de noviembre de 2022

De qué se trata

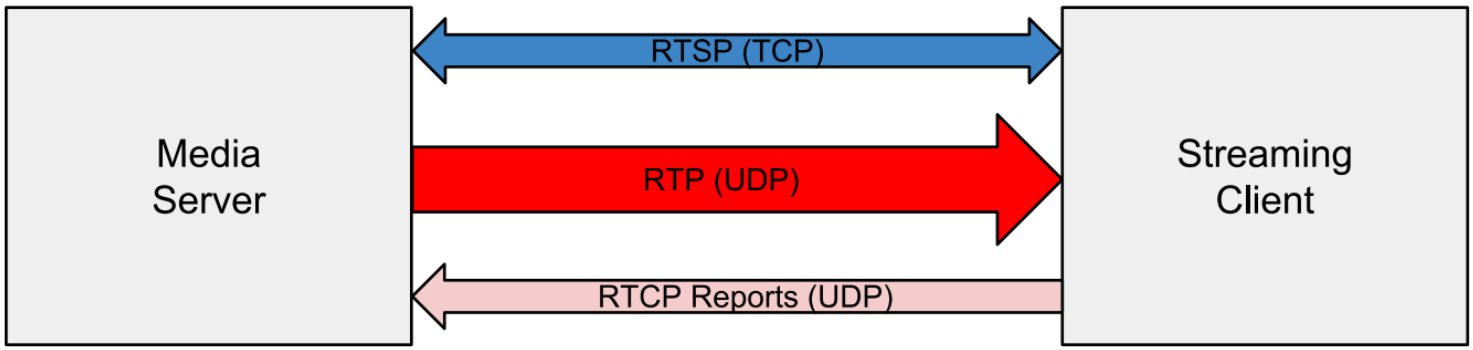
- en 2005 una compañía llamada "Move Networks" introdujo un nuevo paradigma...
  - el contenido multimedia es fragmentado en pequeños "*segments*" o "*chunks*"
  - cada segmento es distribuido mediante HTTP, como cualquier contenido web
  - distintos segmentos con distintas calidades son generados para el **mismo período**
  - el cliente puede decidir qué segmento descargar según sus propios requerimientos
- por lo general cada segmento dura entre 2 y 10 segundos
- es posible utilizar servidores web tradicionales por lo que no se requiere cambios en redes existentes
- ¡rápidamente se convirtió el el paradigma de distribución multimedia dominante!

$\Rightarrow$ También conocido como *HAS* del Inglés "*HTTP Adaptive Streaming*"

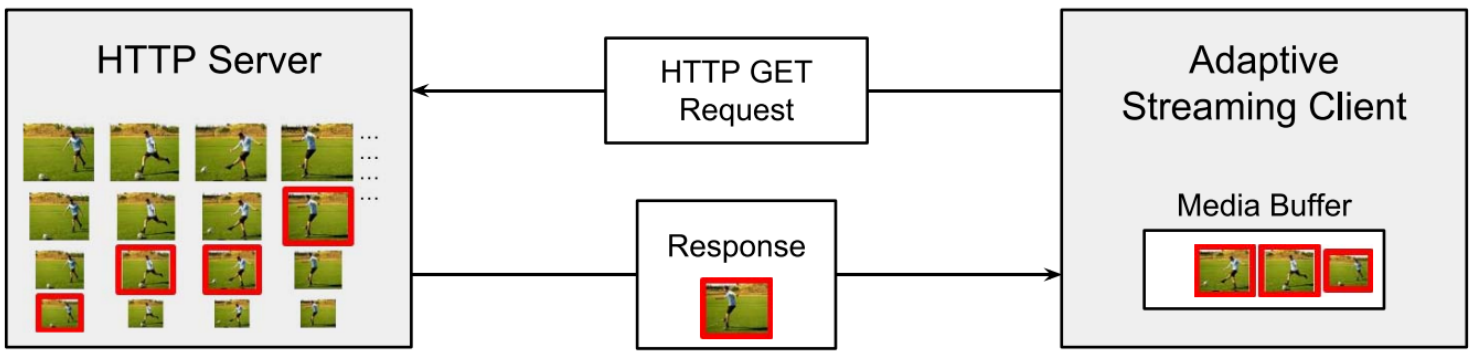
El paradigma anterior

- servidor "empujaba" el tráfico hacia el cliente
  - Real Time Messaging Protocol (RTMP): sobre TCP, orientado a conexión
  - Real Time Transport Protocol (RTP): sobre UDP, no orientado a conexión
- la lógica no estaba en el cliente sino en el servidor
  - Real Time Streaming Protocol (RTSP):
    - en general sobre TCP, orientado a conexión, utiliza el puerto 554
    - similar a HTTP, pero con estados
    - algunas directivas: OPTIONS, DESCRIBE, SETUP, PLAY, PAUSE
  - RTP Control Protocol (RTCP):
    - sobre UDP, no orientado a conexión
    - mediante este protocolo el cliente envía estadísticas al servidor
- el servidor mantiene el estado de todos los clientes
- se utilizan distintos protocolos y servidores dedicados
- el tráfico puede ser bloqueado en algunas redes

RTP vs HAS



(a) Distribución multimedia mediante RTP



(b) Distribución multimedia mediante HAS

Figura: tomada de "A Survey on Bitrate Adaptation Schemes for Streaming Media over HTTP", IEEE Communications Surveys & Tutorials.

$\Rightarrow$ HAS utiliza HTTP en capa de aplicación y TCP en capa de transporte

Push vs Pull delivery

Push-based delivery	***Pull-based delivery***
Lógica en el servidor	Lógica en el cliente
Servidor guarda estado de todas las conexiones	Cliente guarda los estados
Conviven de múltiples protocolos	Sólo se utiliza HTTP
Servidores específicos	Servidores web estándar
Puede ser bloqueado en algunas redes	Las redes en general soportan contenido HTTP

- con "lógica" nos referimos al control de la reproducción y su calidad
- al mantener estados en el servidor, es más probable que el servicio se vea afectado ante caídas parciales
- Servidores web más ampliamente utilizados en el mundo[1]:
  - Apache - 44,3 %
  - nginx - 41,0 %
  - IIS - 8,9 %
  - LiteSpeed Web Server - 3,9 %
  - GWS - 0,9 %

$\Rightarrow$ ¡métodos basados en *push* requieren servidores más complejos, más caros y menos estables!

[1] Fuente: Wikipedia (`https://en.wikipedia.org/wiki/Web_server`), datos al 2019

Algunos conceptos importantes

- **segment** o **chunk**: porción del contenido multimedia, usualmente de 2 a 10 segundos
- **representations**: distintas versiones codificadas del mismo *segment* (pueden cambiar calidades, bitrates, resoluciones)
- **adaptation set**: conjunto de *representations*, por ejemplo video, distintos idiomas de audio, distintos idiomas de subtítulos
- **manifest**: archivo que especifica los distintos *adaptation sets* y *representations* disponibles, y la ruta para obtenerlos

Ejemplo de sesión HAS

(1) el cliente pide al servidor el *manifest* mediante un HTTP GET

(2) en el manifest lee los *adaptation sets* y *representations* disponibles, entre otra información

(3) según algún criterio (en general las preferencias del usuario) se seleccionan los *adaptation sets* a descargar

(4) según algún criterio (en general la de *menor bitrate*) se selecciona la primera *representation* a descargar de cada *adaptation set*

(5) se descarga el primer segmento de las *representations* seleccionadas

(6) a partir de ciertos parámetros (ancho de banda, estado del buffer, batería, CPU, etc.) se selecciona la siguiente *representation* a descargar de cada *adaptation set*

(7) se descarga el siguiente segmento de las *representations* seleccionadas

(8) se vuelve al paso 6 en tanto dure la sesión

Principales estándares

- HTTP Dynamic Streaming (HDS)
  - creador: Adobe
- Smooth Streaming
  - creador: Microsoft
- HTTP Live Streaming (HLS)
  - creador: Apple
- Dynamic Adaptive Streaming over HTTP (DASH)
  - creador: MPEG

$\Rightarrow$ ¡son todos distintas maneras de implementar HAS!

Algo sobre HTTP (1) - introducción

- protocolo de aplicación sin estados que permite transferencia de archivos a través de Internet
- se basa en el modelo cliente-servidor, bajo una lógica de pedido-respuesta (*request-response*)
- el servidor estará esperando por un pedido por parte del cliente, analizará el mensaje y enviará una o más respuestas en consecuencia
- ***resource***: el objetivo de un pedido HTTP, identificado por un *Uniform Resource Identifier* (URI)
- ***representation (HTTP)***: información que representa el estado actual de un *resource*

Cuadro: versiones de HTTP y años de publicación

Versión	Año de publicación
0.9	1991
1.0	1996
1.1	1997
2.0	2015
3.0	2018

Algo sobre HTTP (2) - mensajes y formato

Un mensaje HTTP tiene el siguiente formato:

HTTP-message = *start-line*
*(*header-field* CRLF)
CRLF
[*message-body*]

- *start-line* = *request-line* / *status-line*
- *request-line* = METHOD SP *request-URI* SP *HTTP-version* CRLF
- *status-line* = *HTTP-version* SP *status-code* SP *reason-phrase* CRLF
- *header-field* = *field-name* ":" OWS *field-value* OWS
- *message-body* = *OCTET

Algo sobre HTTP (3) - métodos

- en HTTP los *requests* se realizan en forma de métodos
- cada método indica el propósito por el cuál el cliente ha realizado el *request* y qué espera en respuesta
- una descripción detallada puede ser encontrada en
  - `https://tools.ietf.org/html/rfc7231`
  - `https://tools.ietf.org/html/rfc5789` (PATCH)

Cuadro: Métidos HTTP

Método	**Descripción**
GET	Pide obtener el estado actual de un determinado recurso
HEAD	Igual que GET pero sólo espera encabezado de la respuesta
POST	Pide al recurso que procese los datos enviados en el cuerpo del mensaje
PUT	Pide al recurso que actualice la información a la enviada en el cuerpo del mensaje
DELETE	Pide eliminar toda la información del recurso
CONNECT	Pide establecer un tunel con el servidor identificador por el recurso
OPTIONS	Pide especificar las opciones disponibles para el recurso (métodos soportados)
TRACE	Pide responder el mismo mensaje que fue enviado para dines de diagnóstico
PATCH	Pide al recurso realizar cambios parciales

Algo sobre HTTP (4) - ejemplo: GET

Request

GET /hello.txt HTTP/1.1
User-Agent: curl/7.16.3 libcurl/7.16.3 OpenSSL/0.9.7l zlib/1.2.3
Host: www.example.com
Accept-Language: en, mi

Response

HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
Server: Apache
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
ETag: "34aa387-d-1568eb00"
Accept-Ranges: bytes
Content-Length: 51
Vary: Accept-Encoding
Content-Type: text/plain

¡Hola! Este ejemplo lo saqué del RFC-7230, sección 2.1

HTTP Live Streaming (HLS)

- fue implementado por Apple en 2009
- es uno de los protocolos líderes para la distribución de contenido multimedia mediante HAS
- web oficial `https://developer.apple.com/streaming/`
- la versión 7 (y última) del protocolo está especificada en el RFC-8216

HLS, conceptos importantes

- **master playlist** (*manifest*):
  - un set de *variant streams*
  - su URI debe terminar con m3u8 o m3u
- **media playlist** (*manifest*):
  - refiere a un *variant stream* o una *rendition*
  - especifica cómo obtener los *segments* correspondientes
  - su URI debe terminar con m3u8 o m3u
- **variant stream**:
  - diferentes versiones del mismo contenido, en cuanto a calidad, resolución y bitrate
  - es formado por un conjunto de *renditions* o *rendition groups*
  - son seleccionados debido al entorno (condiciones de red, CPU, buffer, etc.)
- **rendition**:
  - versiones alternativas de un contenido
  - por ejemplo distintos idiomas de audio o subtítulos
  - son seleccionadas según las preferencias del usuario
  - a un set de *renditions* se los llama *rendition groups*

Master Playlist

```
#EXTM3U
#EXT-X-STREAM-INF:BANDWIDTH=150000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/low/index.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=240000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/lo_mid/index.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=440000,RESOLUTION=416x234,CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/hi_mid/index.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=640000,RESOLUTION=640x360,CODECS="avc1.42e00a,mp4a.40.2"
http://example.com/high/index.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=64000,CODECS="mp4a.40.5"
http://example.com/audio/index.m3u8
```

Figura: tomada de `https://developer.apple.com/streaming/`.

- la master playlist es leída una única vez al comienzo de la reproducción, se asume que su contenido no cambia
- tags:
  - EXTM3U:
    - indica que se trata de un archivo M3U extendido
    - todas las *playlists* deben comenzar con este tag
  - EXT-X-STREAM-INF:
    - especifica un *variant stream* que es un conjunto de *renditions*
    - los atributos del tag dan información del *variant stream*
    - la URI siguiente es mandatoria especifica una *rendition* del *variant stream*

Media Playlist - VOD

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:4
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.0,
http://example.com/movie1/fileSequenceA.ts
#EXTINF:10.0,
http://example.com/movie1/fileSequenceB.ts
#EXTINF:10.0,
http://example.com/movie1/fileSequenceC.ts
#EXTINF:9.0,
http://example.com/movie1/fileSequenceD.ts
#EXT-X-ENDLIST
```

Figura: tomada de `https://developer.apple.com/streaming/`.

- para VOD las media playlists son estáticas y especifican las URI de todo el contenido multimedia

Media Playlist - VOD (tags)

- EXT-X-PLAYLIST-TYPE: puede tomar los valores EVENT o VOD
  - puede tomar los valores EVENT o VOD
  - EVENT: el servidor no puede borrar ninguna parte de la *playlist*, sólo puede agregar cosas al final
  - VOD: la *playlist* no debe cambiar nunca
- EXT-X-TARGETDURATION: el tamaño máximo que puede tener un *segment*
- EXT-X-VERSION: indica la versión de la *playlist*
- EXT-X-MEDIA-SEQUENCE: indica el número de secuencia de la primera URI de la lista
- EXTINF: la duración en segundos del *segment* especificado en la URI que está a continuación
- EXT-X-ENDLIST: indica que no se especificarán más *segments* en la *playlist*

Media Playlist - live

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:4
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:10.0,
fileSequence1.ts
#EXTINF:10.0,
fileSequence2.ts
#EXTINF:10.0,
fileSequence3.ts
#EXTINF:10.0,
fileSequence4.ts
#EXTINF:10.0,
fileSequence5.ts
```

Figura: tomada de `https://developer.apple.com/streaming/`.

- puede verse este tipo de *playlists* como ventanas móviles
- el tag EXT-X-ENDLIST no está al final de la lista, indicando que se irán agregando nuevos *segments*
- conforme se van agregando *segments* al final de la lista, se remueven los primeros
- la playlist está constantemente variando su contenido

Media Playlist - event

```
#EXTM3U
#EXT-X-PLAYLIST-TYPE:EVENT
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:4
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:10.00,
fileSequence0.ts
#EXTINF:10.0,
fileSequence1.ts
#EXTINF:10.0,
fileSequence2.ts
#EXTINF:10.0,
fileSequence3.ts
#EXTINF:10.0,
fileSequence4.ts
```

Figura: tomada de `https://developer.apple.com/streaming/`.

- en este tipo de *playlists* el tag EXT-X-PLAYLIST-TYPE toma el valor EVENT
- el tag EXT-X-ENDLIST inicialmente no está al final de la lista, indicando que se irán agregando nuevos *segments*
- no se elimina ningún *segment* anterior, siemplemente se agregan nuevos al final

Tag EXT-X-MEDIA

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="Chinese",URI="chinese/ed.ttml"
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs",NAME="French",URI="french/ed.ttml"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="English",URI="en/chunklist_b160000.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="Spanish",URI="sp/chunklist_b160000.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="Commentary",URI="com/chunklist_b160000.m3u8"
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=2962000,NAME="High",CODECS="avc1.66.30", \
        RESOLUTION=1280x720,AUDIO="aac",SUBTITLES="subs"
1280/chunklist_b2962000.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=1427000,NAME="Medium",CODECS="avc1.66.30", \
        RESOLUTION=768x432,AUDIO="aac",SUBTITLES="subs"
768/chunklist_b1427000.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1,BANDWIDTH=688000,NAME="Low",CODECS="avc1.66.30", \
        RESOLUTION=448x252,AUDIO="aac",SUBTITLES="subs"
448/chunklist_b688000.m3u8
```

- es utilizado para identificar *renditions* alternativas del mismo contenido
- por ejemplo audios y subtítulos de distintos idiomas
- también distintas cámaras grabando el mismo evento

Atributos (1)

Distintos tags definen distintos atributos, siguen algunos ejemplos para el tag **EXT-X-STREAM-INF**:

- BANDWIDTH: el bitrate máximo de *variant stream*
- RESOLUTION: la resolución óptima a la cuál presentar al *variant stream*
- CODECS: una lista separada por coma de los formatos presentes en el *variant stream*

Atributos (2)

Distintos tags definen distintos atributos, siguen algunos ejemplos para el tag **EXT-X-MEDIA**:

- TYPE: puede tomar los valores AUDIO, VIDEO, SUBTITLES o CLOSED-CAPTIONS; es un atributo requerido
- GROUP-ID: especifica el grupo al cual pertenece la *rendition*; es un atributo requerido
- NAME: da una descripción a la *rendition*; es un atributo requerido
- URI: es un atributo opcional (a menos que se trate de subtítulos), especifica especifica la *media playlist* correspondiente a esta *rendition*

Media Segments soportados

- MPEG-2 Transport Stream.
  - Sólo SPTS soportados.
  - Los primeros dos paquetes deben ser PAT y PMT.
- Fragmented MPEG-4.
  - Basados en el ISO Base Media File Format.
  - Cada fragmento corresponde a un *chunk*.
  - Podemos ir pidiendo al archivo de a bytes.
- Packed audio.
  - AAC sobre ADTS.
  - MP3.
  - AC-3.
  - Enhanced AC-3.
- WebVTT
  - Es el único formato de subtítulos soportado.

Show time!

(1) instalar algun servidor web, como por ejemplo nginx

```
sudo apt get install nginx
```

(2) crear una carpeta “hls” en /var/www/html/

```
sudo mkdir /var/www/html/hls
```

(3) instalar ffmpeg

```
sudo apt install ffmpeg
```

(4) crear cada uno de los *variant streams* con el siguiente comando

```
ffmpeg -i <input> -g <gop_size> -s <resolution> \
-aspect <aspect_ratio> -r <frame_rate> -c:v <video_codec> \
-c:a <audio_codec> -b <bitrate> -hls_list_size 0 \
-force_key_frames 'expr:gte(t,n_forced*10)' \
-hls_time <chunk_duration> <output.m3u8>
```

(5) armar la playlist segun los *variant streams* definidos

(6) dejar todo disponible en el servidor web

(7) probar reproducir con VLC en modo *verbose*

```
vlc -vv http://localhost/hls/playlist.m3u8
```

Dynamic Adaptive Streaming over HTTP (MPEG Dash)

- ISO/IEC 23009-1: *Information technology – Dynamic adaptive streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats*.
- Estandarizado por el grupo MPEG en abril de 2012, la última versión revisada es de 2019.
- Es un estándar fomentado por la industria (Microsoft, Apple, Netflix, Qualcomm, Ericsson, Samsung, entre otros) con el objetivo de lograr interoperabilidad.
- Se formó a la vez el *DASH Industry Forum* (`https://dashif.org/`) para promover la adopción del estándar.
- Este foro además desarrolló y mantiene un player de referencia en código abierto (`https://reference.dashif.org/dash.js/`).

Descripción general del sistema

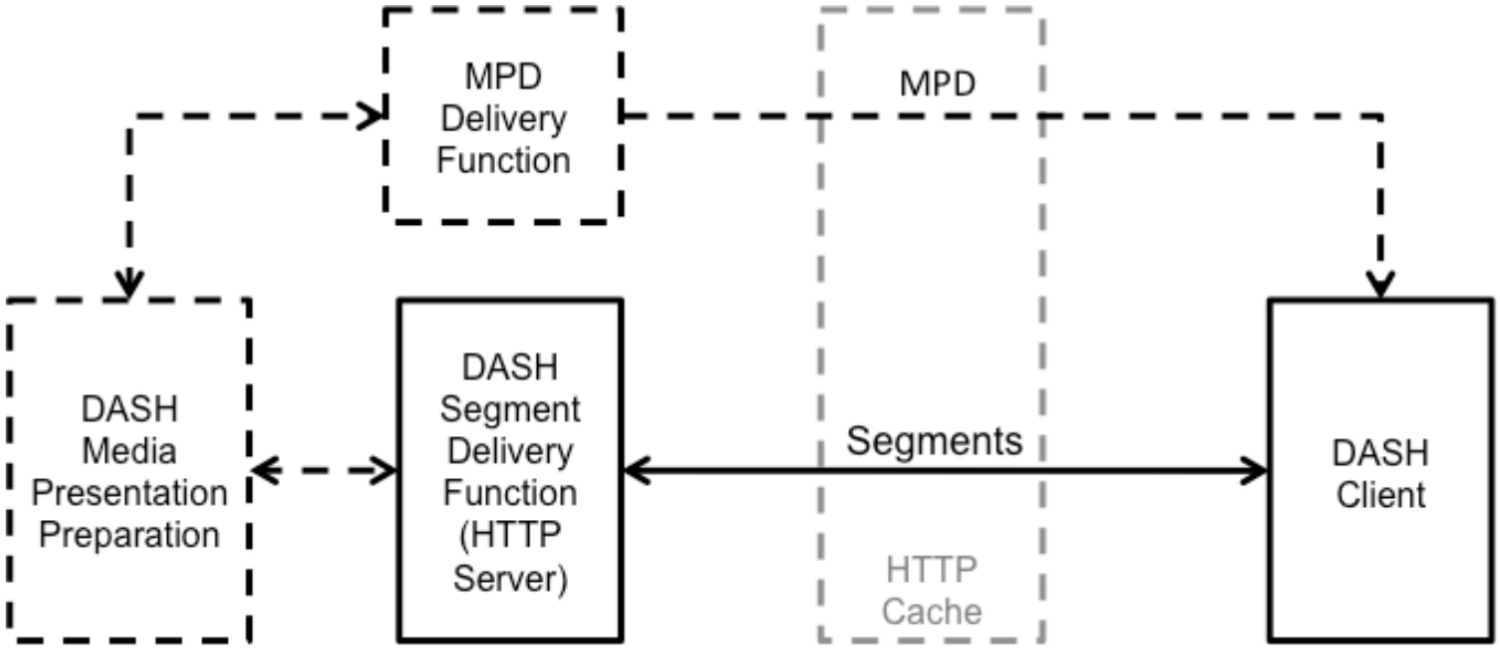


Figura: Tomada del estándar.

- Los Media Presentation Description (MPD) son archivos XML que indican al cliente cómo acceder a cada uno de segmentos (o partes de ellos).
- Todo el control recae principalmente sobre el cliente.

Componentes lógicos de un cliente DASH

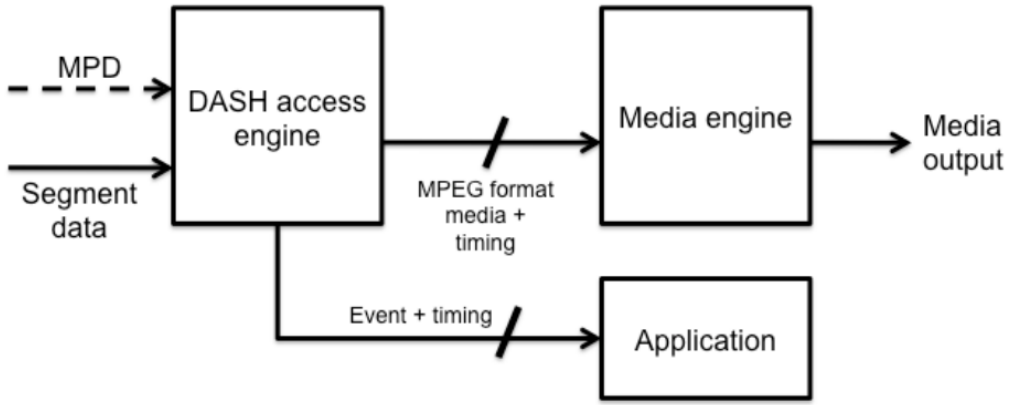


Figura: Tomada del estándar.

- A partir del MPD se piden (y reciben) los segmentos (o partes de ellos).
- El *Media engine* recibe contenido multimedia en dos posibles formatos contentedores: ISO/IEC 14496-12 ISO Base Media File Format, o ISO/IEC 13818-1 MPEG-2 Transport Stream.
- La *Application* recibe información temporal para mapear el contenido multimedia con la línea temporal del archivo MPD.
- Lo anterior es una abstracción. El estándar no especifica cómo debe ser implementado el *player*.

El modelo de datos de DASH

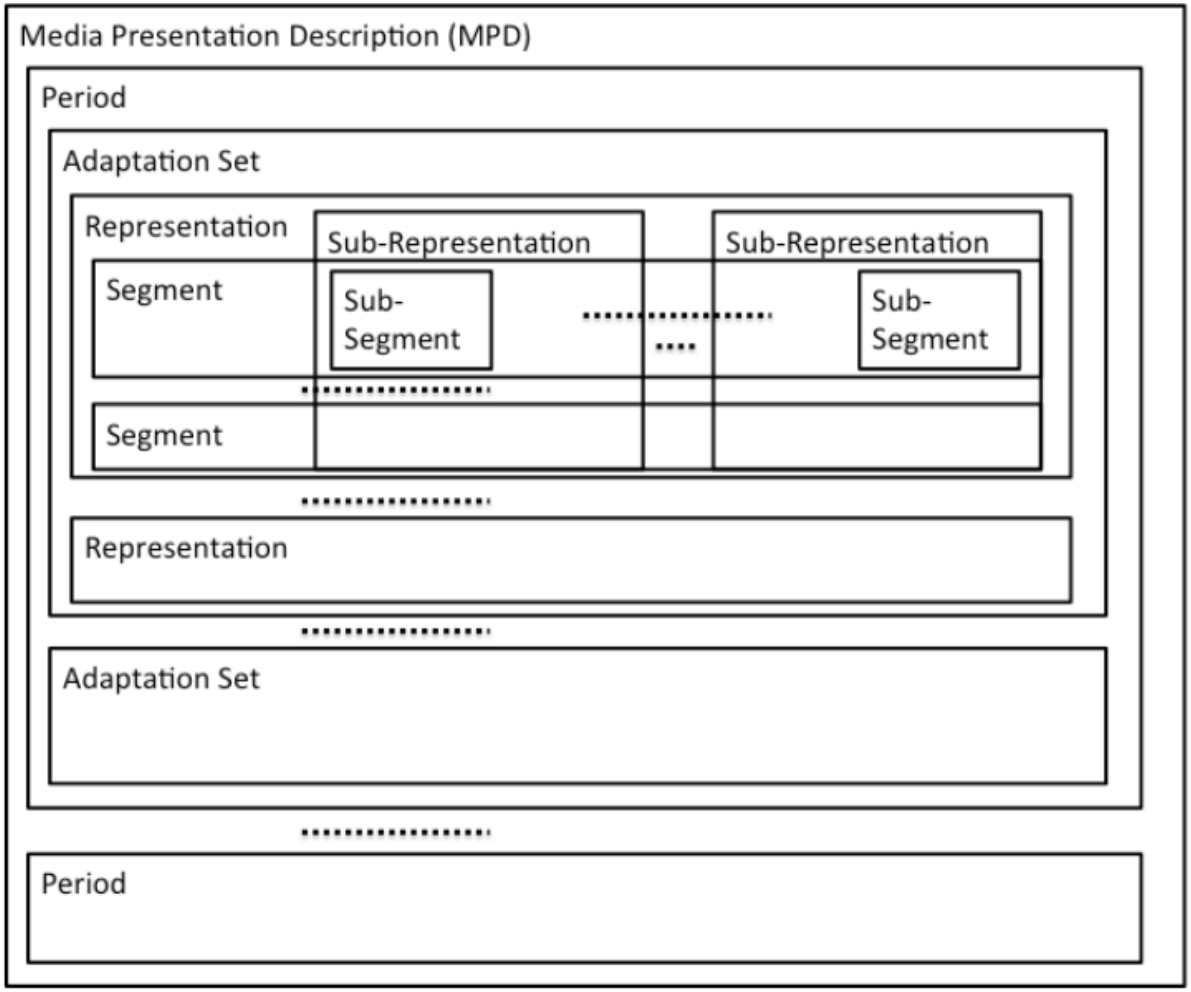


Figura: Tomada del estándar.

Algunas definiciones

- **Media Presentation**: Colección da datos que forman un contenido multimedia.
- **Period**: Intervalo de la *Media Presentation*. Una *Media Presentation* puede estar formada por uno o varios *Periods* contiguos.
- **Media Content Component**: Una media única y continua que puede ser codificada de manera individual.
- **Media Stream**: Versión codificada de un *Media Content Component*.
- **Adaptation Set**: Conjunto de versiones codificadas e intercambiables de un mismo *Media Content Component* (o conjunto de ellos). Conjunto de *Representations* intercambiables.
- **Representation**: Uno o más *Media Streams* listos para ser distribuidos (encapsulados y con *metadata*).
- **Segment**: Unidad de datos asociados a un pedido HTTP. Puede ser un rango de bytes correspondiente a un archivo de mayor tamaño.

Media Presentation Description

```
<!-- MPD Type -->
<xs:complexType name="MPDtype">
  <xs:sequence>
    <xs:element name="ProgramInformation" type="ProgramInformationType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Location" type="xs:anyURI" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Period" type="PeriodType" maxOccurs="unbounded"/>
    <xs:element name="Metrics" type="MetricsType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="EssentialProperty" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="SupplementalProperty" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="UTCTiming" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="id" type="xs:string"/>
  <xs:attribute name="profiles" type="xs:string" use="required"/>
  <xs:attribute name="type" type="PresentationType" default="static"/>
  <xs:attribute name="availabilityStartTime" type="xs:dateTime"/>
  <xs:attribute name="availabilityEndTime" type="xs:dateTime"/>
  <xs:attribute name="publishTime" type="xs:dateTime"/>
  <xs:attribute name="mediaPresentationDuration" type="xs:duration"/>
  <xs:attribute name="minimumUpdatePeriod" type="xs:duration"/>
  <xs:attribute name="minBufferTime" type="xs:duration" use="required"/>
  <xs:attribute name="timeShiftBufferDepth" type="xs:duration"/>
  <xs:attribute name="suggestedPresentationDelay" type="xs:duration"/>
  <xs:attribute name="maxSegmentDuration" type="xs:duration"/>
  <xs:attribute name="maxSubsegmentDuration" type="xs:duration"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

Figura: Tomado de `https://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd`.

Period

```
<!-- Period -->
<xs:complexType name="PeriodType">
  <xs:sequence>
    <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
    <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
    <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
    <xs:element name="AssetIdentifier" type="DescriptorType" minOccurs="0"/>
    <xs:element name="EventStream" type="EventStreamType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="AdaptationSet" type="AdaptationSetType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="Subset" type="SubsetType" minOccurs="0" maxOccurs="unbounded"/>
    <xs:element name="SupplementalProperty" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
  <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute ref="xlink:href"/>
  <xs:attribute ref="xlink:actuate" default="onRequest"/>
  <xs:attribute name="id" type="xs:string"/>
  <xs:attribute name="start" type="xs:duration"/>
  <xs:attribute name="duration" type="xs:duration"/>
  <xs:attribute name="bitstreamSwitching" type="xs:boolean" default="false"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

Figura: Tomado de `https://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd`.

Adaptation Sets

```
<!-- Adaptation Set -->
<xs:complexType name="AdaptationSetType">
  <xs:complexContent>
    <xs:extension base="RepresentationBaseType">
      <xs:sequence>
        <xs:element name="Accessibility" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Role" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Rating" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="Viewpoint" type="DescriptorType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="ContentComponent" type="ContentComponentType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
        <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
        <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
        <xs:element name="Representation" type="RepresentationType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute ref="xlink:href"/>
      <xs:attribute ref="xlink:actuate" default="onRequest"/>
      <xs:attribute name="id" type="xs:unsignedInt"/>
      <xs:attribute name="group" type="xs:unsignedInt"/>
      <xs:attribute name="lang" type="xs:language"/>
      <xs:attribute name="contentType" type="xs:string"/>
      <xs:attribute name="par" type="RatioType"/>
      <xs:attribute name="minBandwidth" type="xs:unsignedInt"/>
      <xs:attribute name="maxBandwidth" type="xs:unsignedInt"/>
      <xs:attribute name="minWidth" type="xs:unsignedInt"/>
      <xs:attribute name="maxWidth" type="xs:unsignedInt"/>
      <xs:attribute name="minHeight" type="xs:unsignedInt"/>
      <xs:attribute name="maxHeight" type="xs:unsignedInt"/>
      <xs:attribute name="minFrameRate" type="FrameRateType"/>
      <xs:attribute name="maxFrameRate" type="FrameRateType"/>
      <xs:attribute name="segmentAlignment" type="ConditionalUintType" default="false"/>
      <xs:attribute name="subsegmentAlignment" type="ConditionalUintType" default="false"/>
      <xs:attribute name="subsegmentStartsWithSAP" type="SAPType" default="0"/>
      <xs:attribute name="bitstreamSwitching" type="xs:boolean"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figura: Tomado de `https://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd`.

Representation

```
<!-- Representation -->
<xs:complexType name="RepresentationType">
  <xs:complexContent>
    <xs:extension base="RepresentationBaseType">
      <xs:sequence>
        <xs:element name="BaseURL" type="BaseURLType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="SubRepresentation" type="SubRepresentationType" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="SegmentBase" type="SegmentBaseType" minOccurs="0"/>
        <xs:element name="SegmentList" type="SegmentListType" minOccurs="0"/>
        <xs:element name="SegmentTemplate" type="SegmentTemplateType" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="id" type="StringNoWhitespaceType" use="required"/>
      <xs:attribute name="bandwidth" type="xs:unsignedInt" use="required"/>
      <xs:attribute name="qualityRanking" type="xs:unsignedInt"/>
      <xs:attribute name="dependencyId" type="StringVectorType"/>
      <xs:attribute name="mediaStreamStructureId" type="StringVectorType"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figura: Tomado de `https://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd`.

- Cada *Representation* es formada por uno o más *Segments*.
- Debe contar con un *Segment* de inicialización, a menos que todos los *Segments* sean autoinicializados.

Segment Base

```
<!-- Segment information base -->
<xs:complexType name="SegmentBaseType">
  <xs:sequence>
    <xs:element name="Initialization" type="URLType" minOccurs="0"/>
    <xs:element name="RepresentationIndex" type="URLType" minOccurs="0"/>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="timescale" type="xs:unsignedInt"/>
  <xs:attribute name="presentationTimeOffset" type="xs:unsignedLong"/>
  <xs:attribute name="indexRange" type="xs:string"/>
  <xs:attribute name="indexRangeExact" type="xs:boolean" default="false"/>
  <xs:attribute name="availabilityTimeOffset" type="xs:double"/>
  <xs:attribute name="availabilityTimeComplete" type="xs:boolean"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>

<!-- Multiple Segment information base -->
<xs:complexType name="MultipleSegmentBaseType">
  <xs:complexContent>
    <xs:extension base="SegmentBaseType">
      <xs:sequence>
        <xs:element name="SegmentTimeline" type="SegmentTimelineType" minOccurs="0"/>
        <xs:element name="BitstreamSwitching" type="URLType" minOccurs="0"/>
      </xs:sequence>
      <xs:attribute name="duration" type="xs:unsignedInt"/>
      <xs:attribute name="startNumber" type="xs:unsignedInt"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Figura: Tomado de `https://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd`.

- Se debe utilizar cuando la *Representation* consta de un único *Segment*.
- En caso de constar de más de un *Segment* deben utilizarse los elementos *Segment List* o *Segment Template*.

Segment List

```
<!-- Segment List -->
<xs:complexType name="SegmentListType">
  <xs:complexContent>
    <xs:extension base="MultipleSegmentBaseType">
      <xs:sequence>
        <xs:element name="SegmentURL" type="SegmentURLType" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute ref="xlink:href"/>
      <xs:attribute ref="xlink:actuate" default="onRequest"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Segment URL  -->
<xs:complexType name="SegmentURLType">
  <xs:sequence>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:attribute name="media" type="xs:anyURI"/>
  <xs:attribute name="mediaRange" type="xs:string"/>
  <xs:attribute name="index" type="xs:anyURI"/>
  <xs:attribute name="indexRange" type="xs:string"/>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

Figura: Tomado de `https://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd`.

- Cada elemento contiene un conjunto de elementos del tipo *SegmentURL*.
- Cada *SegmentURL* referencia a cada uno de los *Segments* consecutivos.

Segment Template

```
<!-- Segment Template -->
<xs:complexType name="SegmentTemplateType">
  <xs:complexContent>
    <xs:extension base="MultipleSegmentBaseType">
      <xs:attribute name="media" type="xs:string"/>
      <xs:attribute name="index" type="xs:string"/>
      <xs:attribute name="initialization" type="xs:string"/>
      <xs:attribute name="bitstreamSwitching" type="xs:string"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<!-- Segment Timeline -->
<xs:complexType name="SegmentTimelineType">
  <xs:sequence>
    <xs:element name="S" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
        <xs:attribute name="t" type="xs:unsignedLong"/>
        <xs:attribute name="n" type="xs:unsignedLong" use="optional"/>
        <xs:attribute name="d" type="xs:unsignedLong" use="required"/>
        <xs:attribute name="r" type="xs:integer" use="optional" default="0"/>
        <xs:anyAttribute namespace="##other" processContents="lax"/>
      </xs:complexType>
    </xs:element>
    <xs:any namespace="##other" processContents="lax" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
  <xs:anyAttribute namespace="##other" processContents="lax"/>
</xs:complexType>
```

Figura: Tomado de `https://standards.iso.org/ittf/PubliclyAvailableStandards/MPEG-DASH_schema_files/DASH-MPD.xsd`.

- Da una manera dinámica para ir obteniendo los distintos *Segments*.

Tipos de *Segments*

- Se especifican cuatro tipos de segmentos:
  - *Initialization Segments*.
  - *Media Segments*.
  - *Index Segments*.
  - *Bitstream Switching Segments*.
- Formatos soportados:
  - ISO Base Media File Format (ISO/IEC 14496-12).
  - MPEG-2 TS (ISO/IEC 13818-1).