Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Proyecto final de carrera:

DASH: Un estándar MPEG para streaming sobre HTTP

Beatriz Casín Núñez

Directora: Silvia Llorente Viejo

Departamento de Arquitectura de Computadores (AC)

Datos del Proyecto

Título: **DASH: Un estándar MPEG para streaming sobre HTTP**

Nombre del estudiante: **Beatriz Casín Núñez**
Titulación: **Ingeniería en Informática Superior**
Créditos: **37,5**
Directora: **Silvia Llorente Viejo**
Departamento: **Arquitectura de Computadores (AC)**

Miembros del tribunal

Presidente: **Jaime M. Delgado Mercé**

Vocal: **Antoni Grau Saldes**

Secretaria: **Silvia Llorente Viejo**

Calificación

Calificación numerica:
Calificación descriptiva:

Fecha:

Índice general

Parte I

Presentación del proyecto

Capítulo 1

Introducción al proyecto

1.1. Introducción

Actualmente el tráfico de streaming en Internet cada día crece más, tal como podemos ver en la figura 1.1, donde este tráfico se corresponde con la leyenda de color rojo (Real-Time Entertainment).

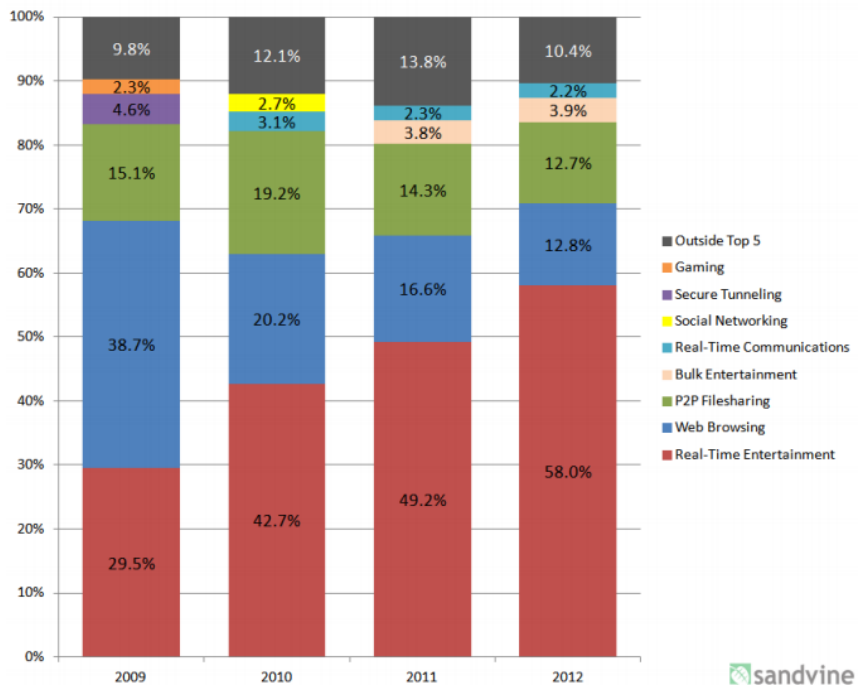


Figura 1.1: Peak Period Aggregate Traffic Composition - North America, Fixed Access[9].

Además el número de dispositivos desde los cuales podemos visualizar contenidos en streaming también ha aumentado en los últimos años (móviles, tablets, smart TV's...), de forma

que desde nuestro móvil podemos ver, por ejemplo, el último capítulo de nuestra serie favorita mientras vamos en el autobús a través de una conexión 3G.

Sin embargo no siempre podemos reproducir todo lo que queremos desde cualquier dispositivo debido a que existen diversos protocolos de streaming, contenido de diferentes formatos etc., y no todos los dispositivos son siempre compatibles. Para resolver estos problemas una posible solución era crear un estándar que permitiese reproducir contenido multimedia en streaming desde cualquier dispositivo, y ese estándar es DASH (Dynamic Adaptive Streaming over HTTP).

1.2. Objetivos

El objetivo principal de este proyecto es realizar una implementación parcial de DASH y para ello se ha desarrollado una aplicación de escritorio en Java con la cual se puede reproducir el contenido multimedia de un MPD (3.1).
Además el usuario debe loguearse en la plataforma MIPAMS (Multimedia Information Protection and Management System) desarrollada por el grupo DMAG (Distributed Multimedia Application Group) y cuando desea reproducir el contenido multimedia de un MPD primero se accede a dicha plataforma y se comprueba que ese usuario tiene permiso para reproducir el contenido de ese MPD.
Una vez que se ha comprobado que el usuario tiene permiso, se parsea el MPD y se obtienen todos los datos necesarios para poder iniciar la reproducción.
Mediante esta aplicación podemos observar durante la reproducción como va cambiando la calidad del contenido multimedia en función del ancho de banda disponible y de la capacidad de CPU del usuario. Esto ocurre si el MPD permite el cambio de calidad automático y el usuario ha seleccionado este modo, ya que también se permite que el usuario escoja una determinada calidad durante toda la reproducción.
Además la aplicación también tiene una opción de simular cambios en el ancho de banda, de forma que se puede comprobar si la calidad realmente cambia correctamente en función de los cambios de la red.

Un segundo objetivo es desarrollar una aplicación para DASH en la plataforma Android, de forma que se pueda reproducir el contenido multimedia de un MPD en un dispositivo Android.
Este segundo objetivo no se ha podido cumplir debido a problemas con los formatos de vídeo en Android. En el capítulo 10 veremos con más detalle estos problemas.

Capítulo 2

Streaming

En este capítulo se realizará una introducción al mundo del streaming, explicando en qué consiste, una breve historia de su evolución y los diferentes protocolos que existen.

2.1. ¿En qué consiste?

El streaming consiste en la distribución contenido multimedia (normalmente vídeo o audio) a través de la red de una forma continua donde el usuario puede ir visualizando dicho contenido a medida que se va descargando.

2.2. Evolución

En los años 80 la informática empezó a llegar al *público de a pié*. Sin embargo los ordenadores de aquella época no eran lo suficientemente potentes y pocos eran capaces de reproducir vídeo y audio de alta calidad.
Unos años más tarde, en la década de los 90, la mayoría de ordenadores ya eran capaces de reproducir archivos multimedia. Sin embargo todavía hacía falta mejorar otros aspectos como el ancho de banda, ya que la simple descarga de un fichero MP3 de unos cinco minutos de duración podía tardar horas, por tanto intentar reproducir este fichero en streaming era prácticamente imposible, y había que descargase el fichero entero para poder reproducirlo.
A pesar de esto en la década de los 90 se emitieron los primeros conciertos en directo a través de Internet. El primer grupo fue Severe Tire Damage, el 24 de junio de 1993 y su concierto pudo verse en todo el mundo gracias a Mbone[1], y un año más tarde lo hicieron los Rolling Stones.
Un año más tarde, en 1995, Real Networks emitió el primer evento deportivo a través de

[1]MBone(IP Multicast Backbone): red virtual sobre Internet, creada por Van Jacobson, Steve Deering y Stephen Casner en 1992, que utilizando técnicas de transmisión multicast permite, entre otras aplicaciones, la transmisión de videoconferencias a gran escala optimizando el uso de recursos.

Internet, un partido de béisbol entre los Yankees y los Seatle Mariners y dos años más tarde lanzaron RealPlayer, el primer sistema de reproducción de vídeo en streaming, basado en la tecnología que habían utilizado en la reproducción del partido de béisbol. En 1999 Apple introdujo un formato de streaming multimedia en su aplicación QuickTime 4. Alrededor de 2002 se desarrolló un formato de vídeo para streaming para reproductores basados en Flash.
En 2005 se lanzó YouTube, un sitio web donde los usuarios pueden subir y compartir vídeos, que utiliza un reproductor en línea basado en Adobe Flash para servir su contenido, aunque actualmente también puede utilizar un reproductor basado en HTML5. En la figura 2.1 podemos ver una infografía[2] que muestra algunos datos interesantes sobre YouTube.

Años más tarde, en 2008, apareció la aplicación Spotify, que permite reproducir música vía streaming. A fecha de diciembre de 2012, Spotify cuenta con más de 20 millones de usuarios registrados en los países disponibles, de los cuales más de 5 millones son de pago. Finalmente, destacar que actualmente existen algunas plataformas como Netflix que permite ver series y películas de forma legal vía streaming y que a mediados de 2011 ya contaba con más de 25 millones de suscriptores en Estados Unidos y Canadá.

2.3. Protocolos

A continuación se describirán brevemente algunos de los principales protocolos de streaming.

2.3.1. Real-time Trasport Protocol (RTP)

Protocolo del nivel de aplicación (sesión en la pila OSI) para la transmisión de información en tiempo real. Se utiliza junto al protocolo RTCP (2.3.3) y funciona sobre UDP, por lo tanto no garantiza que todos los paquetes lleguen a su destino. También se puede utilizar junto al protocolo RTSP(2.3.2).Se considera como el principal estándar para el transporte de audio y vídeo en las redes IP. Funciona tanto en multicast como en unicast.
La información que proporciona incluye timestamps (para la sincronización), números de secuencia (para la pérdida de paquetes y la detección de reordenamiento) y el formato del payload que indica el formato de codificación de los datos. Establece una sesión para cada stream multimedia, donde una sesión consiste en una dirección IP con un par de puertos para RTP y RTCP.

[2]Infografía: representación gráfica visual que pretende presentar información compleja de una forma rápida y clara.

Figura 2.1: Infografía sobre YouTube[8].

2.3.2. Real Time Streaming Protocol (RTSP)

Protocolo del nivel de aplicación para el control de la entrega de datos en tiempo real que permite mantener una o múltiples sesiones. Actúa como un control remoto en red de los servidores multimedia.
Su operación y sintaxis son similares a HTTP sin embargo se diferencia en algunos aspectos:

- Los datos son transportados en un protocolo diferente.
- El servidor necesita mantener el estado de la conexión, a diferencia de HTTP.
- Tanto el servidor como el cliente pueden lanzar peticiones.
- Utiliza rtsp:// en lugar de http://

Algunas de sus propiedades son:

- **Extensible:** Se pueden añadir nuevos parámetros y métodos.
- **Seguro:** Utiliza métodos de autentificación HTTP y reutiliza mecanismos de seguridad web.
- **Independiente del protocolo de transporte:** Puede utilizar tanto TCP como UDP.
- **Capacidad multiservidor:** En una presentación los streams multimedia pueden estar en servidores diferentes. El cliente establece automáticamente varias sesiones concurrentes de control con los diferentes servidores y la capa de transporte se encarga de la sincronización.
- **Control de dispositivos de grabación:** Permite controlar tanto dispositivos de reproducción como de grabación.

Las principales peticiones RTSP son:

- **DESCRIBE:** Obtiene la descripción del objeto que se encuentra en la URL RTSP. Esta descripción contiene, entre otros datos, la lista de los streams multimedia que serán necesarios en la reproducción. Esta petición forma parte de la fase de inicialización.
- **SETUP:** Especifica como debe ser transportado el flujo de datos. La petición contiene la URL del stream multimedia y un especificador de transporte.
  La respuesta del servidor normalmente confirma los parámetros escogidos y añade las partes no completadas. Se debe configurar cada stream con SETUP antes de enviar una petición de PLAY.
- **PLAY:** Esta petición hará que el servidor empiece a enviar los datos de los streams especificados.

- **PAUSE:** Detiene temporalmente la reproducción de uno o más streams. Posteriormente se puede reanudar la reproducción mediante la petición PLAY.
- **TEARDOWN:** Se utiliza para finalizar la sesión. Se detienen los streams y se liberan los recursos asociados.

En la figura 2.2 podemos ver los estados por los que pasa el cliente al utilizar las peticiones anteriores.

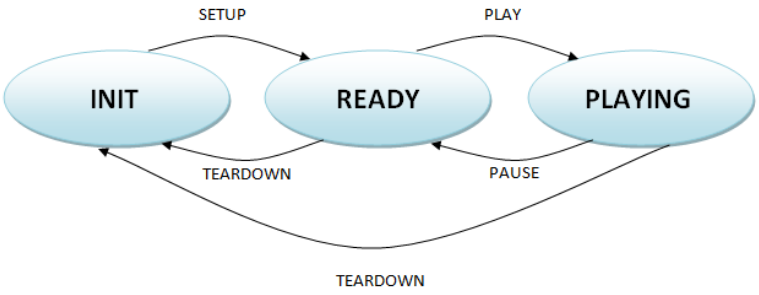


Figura 2.2: Diagrama de estados del cliente en una sesión RTSP.

2.3.3. RTP Control Protocol (RTCP)

Protocolo que proporciona información de control para un flujo RTP(2.3.1). Su función principal es informar de la calidad del servicio (QoS) proporcionado por RTP.
RTCP define varios tipos de paquetes:

- **Sender Report (SR):** Se envía periódicamente por parte del emisor activo para informar de las estadísticas de transmisión y recepción de los paquetes RTP enviados durante un intervalo.
- **Receiver Report (RR):** Lo utilizan los participantes pasivos para enviar estadísticas sobre la recepción.
- **Source Description (SDES):** Se utiliza para enviar el CNAME[3] a los participantes. También se utiliza para enviar otra información como el nombre, correo electrónico, número de teléfono y dirección del emisor.
- **End of participation(BYE):** Se utiliza para indicar el fin de la participación en una sesión.
- **Application-specific message (APP):** Proporciona un mecanismo para diseñar extensiones específicas de la aplicación para el protocolo RTCP.

[3]CNAME (Canonical Name): es un tipo de *resource record* en el DNS que especifica que el nombre del dominio es un alias de otro, *canonical domain name*

2.4. HTTP Streaming

Después de ver algunos de los protocolos de streaming más utilizados hay que tener en cuenta algunos inconvenientes que existen.
Actualmente en Internet existen muchas redes de distribución de contenidos (CDN, Content Delivery Network) y muchas no soportan RTP. Además algunos firewalls tampoco admiten este tipo de paquetes. Otro inconveniente es que en RTP o RTMP (2.5.1) el servidor tiene que gestionar una sesión diferente por cada cliente. También, tal como hemos visto, algunos de los protocolos no funcionan por sí solos y necesitan utilizarse junto a otros como en el caso de RTP, que funciona junto con RTCP y también junto con RTSP, por tanto esto incrementa la información que se necesita transmitir a través de la red.
Una solución a estos inconvenientes es utilizar HTTP Streaming, ya que la gran mayoría de firewalls admiten este tipo de tráfico, lo mismo que las CDNs (que además permiten reducir el tráfico de larga distancia), y el cliente no necesita mantener una sesión en el servidor.
Por estos motivos ha aumentado la popularidad del HTTP streaming y se han creado diferentes implementaciones como HTTP Live Streaming (HLS) de Apple, Microsoft Smooth Streaming y Adobe HTTP Dynamic Streaming. Sin embargo cada una de ellas utiliza diferentes formatos en sus segmentos y sus ficheros manifest, por lo que si un dispositivo quisiese reproducir contenido de un servidor de cada plataforma debería soportar el correspondiente protocolo cliente propietario, lo cual es un inconveniente.

2.5. Adaptive Streaming

Como acabamos de ver el HTTP streaming parece una buena solución, pero se puede mejorar utilizando la técnica de Adaptive Streaming. Esta técnica consiste en escoger la calidad del contenido a reproducir en función del ancho de banda disponible y de la capacidad de CPU del cliente. Para ello se deberá codificar el fichero con diferentes bitrates y posteriormente se deberán segmentar en segmentos de unos pocos segundos. Al iniciar la reproducción se escoge una calidad y durante la reproducción el cliente le enviará información al servidor de forma que pueda mejorar o empeorar la calidad de la reproducción.
Algunas implementaciones de Adaptive Streaming son Adobe HTTP Dynamic Streaming, Adobe RTMP Dynamic Streaming , HTTP Live Streaming (HLS) de Apple y Microsoft Smooth Streaming. Como podemos ver, casi todas estas implementaciones son las mismas que hemos visto en 2.4 excepto Adobe RTMP Dynamic Streaming. A continuación hablaremos del protocolo RTMP.

2.5.1. Real Time Messaging Protocol (RTMP)

Protocolo inicialmente propietario desarrollado por Macromedia para el streaming de audio, de vídeo y de datos a través de Internet, entre un reproductor Flash y un servidor.

Posteriormente Adobe adquirió Macromedia y en el 2009 publicó una versión incompleta de la especificación del protocolo para uso público. Está basado en TCP y mantiene conexiones persistentes
Para enviar los streams de forma fluida y de forma que se transmita tanta información como sea posible, los streams se dividen en fragmentos y su tamaño se negocia dinámicamente entre el cliente y el servidor, aunque en ocasiones se mantiene sin cambios. Por defecto el tamaño determinado para los fragmentos de audio es de 64 bytes y el de los fragmentos de vídeo y otros tipos de datos, de 128 bytes. Los fragmentos de diferentes streams se pueden intercalar y multiplexar en una sola conexión.

Después de establecer una conexión TCP, se establece una conexión RTMP realizando inicialmente un proceso de handshaking entre el cliente y el servidor, tal como se muestra en la figura 2.3.
Después el cliente y el servidor pueden negociar una conexión intercambiando mensajes AMF[4] codificados. Si este proceso finaliza con éxito, el cliente envía al servidor un mensaje "createStream", seguido de un ping y seguido de un mensaje "play". El servidor responderá con una serie de comandos "onStatus" seguidos por los datos del stream encapsulados en mensajes RTMP.

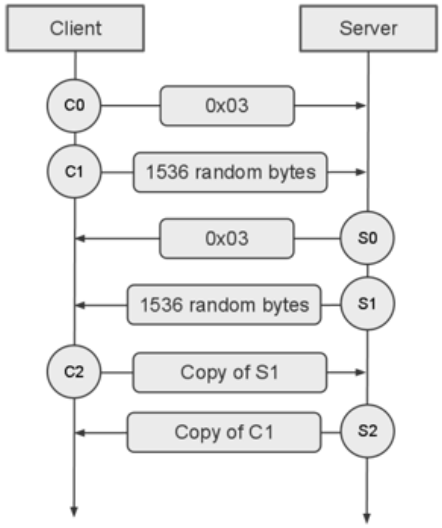


Figura 2.3: Diagrama del proceso de handshake en RTMP[22].

[4]AMF (Action Message Format): formato binario utilizado para enviar mensajes entre un cliente Adobe Flash y un servicio remoto, normalmente un Flash Media Server.

Capítulo 3

Dynamic Adaptive Streaming over HTTP

Dynamic Adaptive Streaming over HTTP (DASH) es un estándar para streaming adaptativo sobre HTTP desarrollado por MPEG(Moving Picture Expert Group) que se convirtió en estándar internacional en noviembre de 2011 y fue publicado como ISO/IEC 23009-1:2012[14] en abril de 2012.
En el modelo de streaming que define, el control recae exclusivamente en el cliente, el cual solicita la información mediante el protocolo HTTP a servidores web estándar. Por tanto este estándar se centra en el formato de los datos utilizados.
DASH principalmente define dos formatos:

- Media Presentation Description (MPD) (3.1): Fichero de metadatos que describe el contenido disponible y otras características.
- Formato de los segmentos (3.2).

En la figura 3.1 podemos ver un escenario simple de streaming entre un servidor HTTP y un cliente DASH, donde el formato y las funcionalidades de las cajas rojas se definen en el estándar. El resto no entran dentro del ámbito del estándar.
En esta figura podemos observar que el contenido multimedia es almacenado en un servidor HTTP y se entrega al cliente mediante este protocolo.
El proceso que debe realizar el cliente para reproducir el contenido multimedia es el siguiente:

- Primero obtiene el MPD mediante HTTP, correo electrónico o otros tipos de transporte.
- Posteriormente lo parsea para obtener la información necesaria para la reproducción.
- Una vez obtenida la información, selecciona las características adecuadas y empieza a realizar el streaming obteniendo los segmentos mediante peticiones HTTP.

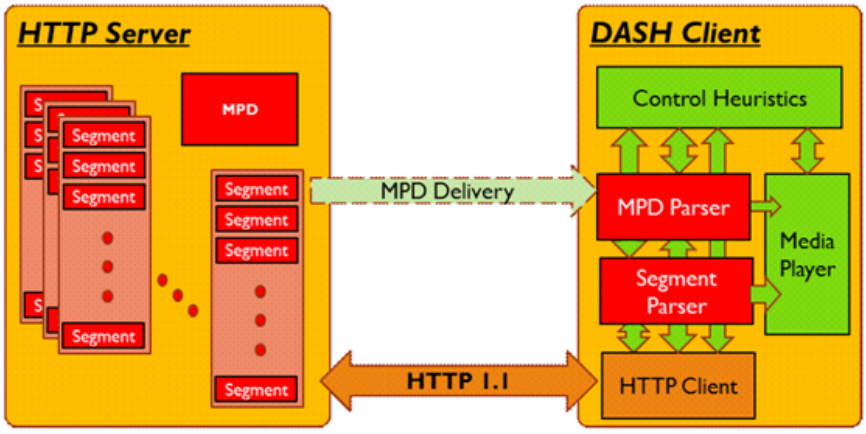


Figura 3.1: Escenario DASH [20].

- Durante la petición de segmentos, monitoriza los cambios en el ancho de banda de la red y decide si es necesario realizar algún cambio.

Finalmente, destacar que DASH soporta tanto streaming bajo demanda como streaming en vivo. En el segundo caso, el MPD se actualizará cada cierto tiempo y el cliente tendrá que ir obteniendo estas actualizaciones.

3.1. Media Presentation Description (MPD)

Media Presentation Description (MPD) es un documento XML que contiene los metadatos necesarios para un cliente DASH de forma que pueda acceder a los segmentos y proporcionar un servicio de streaming al usuario.

En la figura 3.2 podemos ver el modelo de datos jerárquico de un MPD, que consiste en:

- Una secuencia de uno o más Periods (3.1.1).
- Cada Period contiene uno o más Adaptation Sets (3.1.2).
- Cada Adaptation Set contiene una o más Representations (3.1.3).
- Cada Representation puede contener una o más Sub-Representations
- Adaptation Sets, Representations y Sub-Representations comparten atributos y elementos comunes

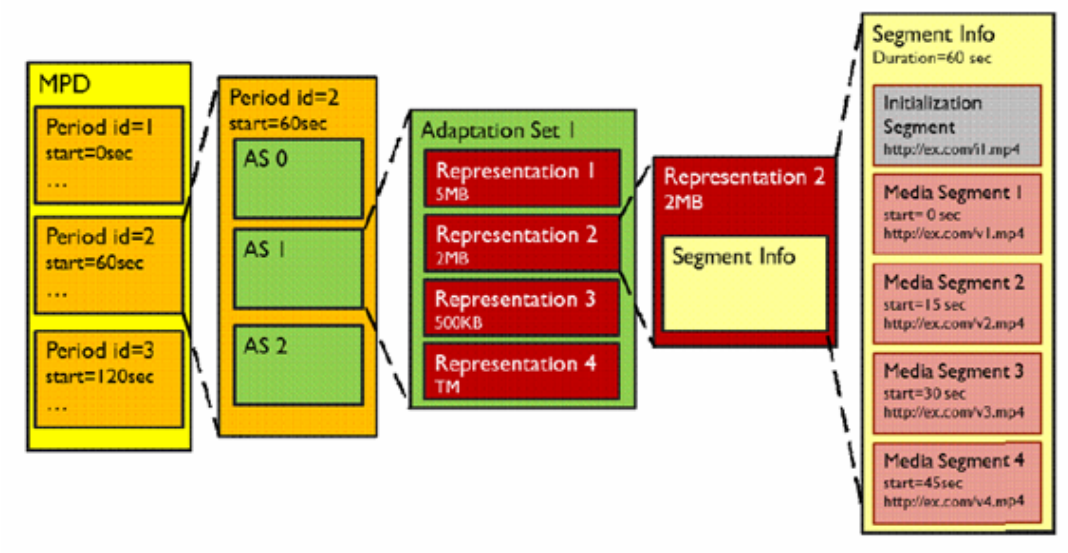


Figura 3.2: Modelo de datos jerárquico de un MPD [20].

- Cada Period puede contener uno o más Subsets que restringen la combinación de Adaptation Sets en la reproducción del contenido multimedia.
- Cada Representation consiste en uno o más Segments (3.1.4).
- Cada Segment consiste en uno o más Subsegments.

En la tabla 3.1 podemos ver algunos de los principales elementos y atributos que puede contener el elemento MPD.

Nombre del elemento o atributo	**Uso**	**Descripción**
MPD		Elemento raíz
@id	OP	Especifica un identificador para el MPD. Si no se especifica se puede utilizar, por ejemplo, la URL del MPD como identificador.
@profiles	OB	Especifica una lista de Media Presentation profiles (3.3).
@type	OPD default: static	Especifica si el MPD debe ser actualizado (@type="dynamic") o no (@type="static").

@availabilityStartTime	COB Obligatorio para @type= “dynamic”	En el caso de que @type=“dynamic” este atributo debe estar presente. En este caso especifica la base para el cálculo del tiempo (en UTC) en el que está disponible cualquier segmento del MPD. En el caso de que @type=“static” este atributo especifica el tiempo en el que están disponibles todos los segmentos del MPD. Si el atributo no aparece entonces todos los segmentos del MPD están disponibles en el momento en el que está disponible el MPD.
@availabilityEndTime	OP	Especifica la última disponibilidad del tiempo final de un segmento para cualquier segmento del MPD. Si este atributo no está presente, su valor es desconocido.
@mediaPresentationDuration	COB Obligatorio para type= “static”	Especifica la duración de todo el MPD. Debe estar presente en el caso de que el atributo **MPD**@minimumUpdatePeriod no aparece.
@minimumUpdatePeriod	OP	Especifica el periodo más pequeño entre posibles cambios en el MPD. Si no aparece significa que el MPD no cambia.
@minBufferTime	OB	Especifica una duración que se utiliza para saber cuando se puede iniciar la reproducción.
ProgramInformation	0..N	Contiene información descriptiva sobre el contenido multimedia.
BaseURL	0..N	Contiene una Base URL que puede ser utilizada para resolver todas las URLs del MPD.
Location	0..N	Especifica una ubicación donde el MPD está disponible.
Period	1..N	Especifica la información de un Period (3.1.1).
Leyenda Para los atributos: OB = obligatorio, OP: Opcional, OPD= opcional con valor por defecto, COB=condicionalmente obligatorio Para los elementos: <mínimo>...<máximo> (N = ilimitado) Los elementos aparecen en negrita y los atributos van precedidos de @		

Tabla 3.1: Semántica del elemento MPD

A continuación podemos ver un ejemplo del elemento MPD.

```
<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"
  profiles="urn:mpeg:dash:profile:isoff-main:2011"
  type="static"
  mediaPresentationDuration="PT0H9M56.46S"
  minBufferTime="PT15.0S">
  <BaseURL>
    http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/bunny_15s/
  </BaseURL>
  <Period start="PT0S">
```

3.1.1. Period

El MPD consta de varios Periods, donde un Period es un intervalo del contenido multimedia en el eje del tiempo y se representa mediante un elemento Period. Cada Period tiene un tiempo de inicio y una duración y consta de uno o más Adaptation Sets.

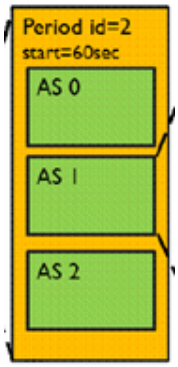


Figura 3.3: Estructura de un Period

El tiempo de inicio de un Period (PeriodStart) se puede determinar de varias formas:

- Mediante el atributo @start.
- Si el atributo @start no existe, pero el Period anterior contiene el atributo @duration, PeriodStart será la suma de el PeriodStart anterior más el valor de @duration del Period anterior.
- Si el atributo @start no existe pero el Period es el primero del MPD y el atributo @type del elemento MPD es 'static', entonces PeriodStart es 0.

En el caso de que no exista el atributo @start, ni el Period anterior contenga el atributo @duration o sea el primero del MPD y el valor del atributo @type sea 'dynamic' entonces se considera al Period como un Early Available Period. Este tipo de Periods se pueden utilizar para anunciar algunos datos no-multimedia antes de que los datos multimedia estén disponibles, por tanto este Period no está disponible para la reproducción. Cuando se conoce el valor de PeriodStart, en una actualización del MPD, entonces el Period pasa a ser un Period normal.

En la tabla 3.2 podemos ver algunos de los principales elementos y atributos que puede contener el elemento Period.

Nombre del elemento o atributo	**Uso**	**Descripción**
Period		Especifica la información del Period
@xlink:href	OP	Especifica una referencia a un elemento Period externo.
@id	OP	Especifica un identificador para el Period, que debe de ser único dentro del MPD.
@start	OP	Especifica el PeriodStart del Period.
@duration	OP	Especifica la duración del Period.
@bitstreamSwitching	OPD default: false	Si su valor es 'true' es equivalente a que el atributo @bitstreamSwitching de cada AdaptationSet dentro del Period sea 'true'.
BaseURL	0..N	Contiene una Base URL que puede ser utilizada para resolver las URLs del Period.
SegmentBase	0..1	Especifica la información de Segment Base. Se sobreescribe por la información de AdaptationSet.SegmentBase y Representation.SegmentBase, en el caso de que existan estos elementos.
SegmentList	0..1	Especifica la información de Segment List. Se sobreescribe por la información de AdaptationSet.SegmentList y Representation.SegmentList, en el caso de que existan estos elementos.
SegmentTemplate	0..1	Especifica la información de Segment Template. Se sobreescribe por la información de AdaptationSet.SegmentTemplate y Representation.SegmentTemplate, en el caso de que existan estos elementos.
AdaptationSet	0..N	Especifica un AdaptationSet.
Subset	0..N	Especifica un Subset.
Leyenda Para los atributos: OB = obligatorio, OP: Opcional, OPD= opcional con valor por defecto, COB=condicionalmente obligatorio Para los elementos: <mínimo>...<máximo> (N = ilimitado) Los elementos aparecen en negrita y los atributos van precedidos de @		

Tabla 3.2: Semántica del elemento Period

3.1.2. Adaptation Set

Cada Period consta de uno o más Adaptation Sets, donde un Adaptation Set proporciona información sobre uno o más componentes multimedia y sus diferentes alternativas de codificación y se representa mediante un elemento AdaptationSet.
Cada Adaptation Set consta de una o más Representations donde únicamente una de ellas puede ser reproducida en un momento determinado. Todas las Representations del mismo Adaptation Set representan el mismo contenido multimedia, por lo tanto contiene streams multimedia que se consideran prácticamente equivalentes.

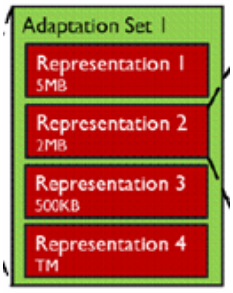


Figura 3.4: Estructura de un Adaptation Set

En la tabla 3.3 podemos ver algunos de los principales elementos y atributos que puede contener el elemento AdaptationSet.

Nombre del elemento o atributo	**Uso**	**Descripción**
AdaptationSet		Descripción del Adaptation Set.
@xlink:href	OP	Especifica una referencia a un elemento AdaptationSet externo.
@id	OP	Especifica un identificador para el AdaptationSet, que debe de ser único dentro del Period.
CommonAttributesElements		Especifica los atributos y los elementos comunes.
@contentType	OP	Especifica el tipo del componente del contenido multimedia para el Adaptation Set actual.
@bitstreamSwitching	OP	Si su valor es true indica que se puede cambiar de Representation dentro del mismo Adaptation Set durante la presentación
ContentComponent	0..N	Especifica las propiedades de un componente del contenido multimedia contenido en el Adaptation Set actual.
BaseURL	0..N	Contiene una Base URL que puede ser utilizada para resolver las URLs del Adaptation Set.

SegmentBase	0..1	Especifica la información de Segment Base. Se sobreescribe por la información de Representation.SegmentBase, en el caso de que exista este elemento.
SegmentList	0..1	Especifica la información de Segment List. Se sobreescribe por la información de Representation.SegmentList, en el caso de que exista este elemento.
SegmentTemplate	0..1	Especifica la información de Segment Template. Se sobreescribe por la información de Representation.SegmentTemplate, en el caso de que exista este elemento.
Representation	0..N	Especifica una Representation.
Leyenda Para los atributos: OB = obligatorio, OP: Opcional, OPD= opcional con valor por defecto, COB=condicionalmente obligatorio Para los elementos: <mínimo>...<máximo> (N = ilimitado) Los elementos aparecen en negrita y los atributos van precedidos de @. Las listas de elementos y atributos aparecen en negrita y cursiva.		

Tabla 3.3: Semántica del elemento AdaptationSet

3.1.3. Representation

Cada Adaptation Set consta de una o más Representations, donde una Representation es una alternativa de codificación de un componente multimedia, que se diferencia de otras Representations del mismo Adaptation Set en el bitrate, resolución, número de canales u otras características. Se representa mediante el elemento Representation.
Cada Representation consta de uno o más Segments y empieza al inicio del Period (PeriodStart) al que pertenece y acaba al final del mismo.
En la tabla 3.4 podemos ver algunos de los principales elementos y atributos que puede contener el elemento Representation.

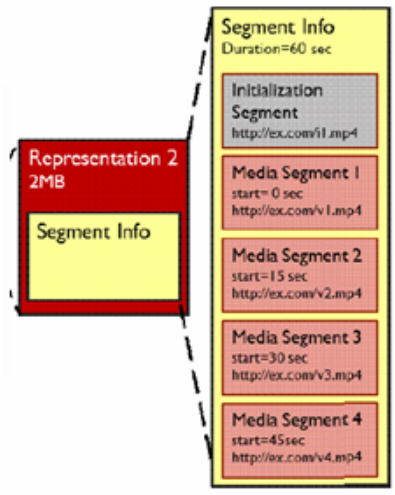


Figura 3.5: Estructura de una Representation

Nombre del elemento o atributo	**Uso**	**Descripción**
AdaptationSet		Descripción de la Representation.
@id	OB	Especifica un identificador para la Representation, que debe de ser único dentro del Period, a no ser que la Representation sea funcionalmente idéntica a otra dentro del mismo Period.
@bandwidth	OB	Bandwidth mínimo necesario.
CommonAttributesElements		Especifica los atributos y los elementos comunes.
BaseURL	0..N	Contiene una Base URL que puede ser utilizada para resolver las URLs de la Representation.
SubRepresentation	0..N	Especifica la información sobre una SubRepresentation que se encuentra dentro de una Representation.
SegmentBase	0..1	Especifica la información de Segment Base.
SegmentList	0..1	Especifica la información de Segment List.
SegmentTemplate	0..1	Especifica la información de Segment Template.
Leyenda Para los atributos: OB = obligatorio, OP: Opcional, OPD= opcional con valor por defecto, COB=condicionalmente obligatorio Para los elementos: <mínimo>...<máximo> (N = ilimitado) Los elementos aparecen en negrita y los atributos van precedidos de @. Las listas de elementos y atributos aparecen en negrita y cursiva.		

Tabla 3.4: Semántica del elemento Representation

3.1.4. Segments

Cada Representation consta de uno o más Segments, donde un Segment es una de las partes en la que se ha dividido un stream multimedia.
Un Segment contiene una URI, que apunta a un contenido en un servidor que puede ser descargado mediante una petición HTTP-GET, pudiendo indicar también un rango de bytes.
Una Representation contiene *Segment Information* mediante los elementos BaseURL, SegmentBase, SegmentTemplate y/o SegmentList, que proporciona información sobre la ubicación, la disponibilidad y las propiedades de los Segments de una Representation.

Para especificar esta información cada Representation utilizará una de las siguientes alternativas:

- Uno o más elementos SegmentList (3.1.4)
- Un elemento SegmentTemplate (3.1.4)
- Uno más elementos BaseURL, como mucho un elemento SegmentBase (3.1.4) y ningún elemento SegmentTemplate o SegmentList.

Tanto SegmentBase, como SegmentTemplate y SegmentList comparten elementos comunes.

Segment Base

El elemento SegmentBase es suficiente en el caso de que únicamente haya un único segmento de stream multimedia en la Representation. En el caso de que haya múltiples, se debe utilizar el elemento SegmentList o SegmentTemplate.

Segment List

Una Segment List consta de uno o más elementos SegmentList, cada uno de los cuales contiene una lista de elementos SegmentURL, los cuales contienen las URLs de los segmentos de stream multimedia y también pueden contener un rango de bytes y un Index Segment.
A continuación podemos ver un ejemplo del elemento SegmentList.

```
<SegmentList duration="15">
  <SegmentURL media="bunny_15s_50kbit/bunny_15s1.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s2.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s3.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s4.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s5.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s6.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s7.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s8.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s9.m4s"/>
  <SegmentURL media="bunny_15s_50kbit/bunny_15s10.m4s"/>
                    . . .
```

Segment Template

Un Segment Template se representa con el elemento SegmentTemplate. Este elemento utiliza identificadores a los cuales se les asigna valores dinámicamente para crear una lista de segmentos.
En la tabla(3.5) podemos ver los identificadores que pueden aparecer en los atributos de SegmentTemplate y el parámetro por el cual deben de ser sustituidos.
A continuación podemos ver un ejemplo del elemento SegmentTemplate.

```
<SegmentTemplate
  timescale="1000" duration="10000"
  media="mp4-main-ogop-$RepresentationID$$Number$.m4s"
  startNumber="1"
  initialization="mp4-main-ogop-mpd-V-BS_init.mp4"/>
  <Representation
    id="h264ogop_low" mimeType="video/mp4"
    codecs="avc1.64000d" width="320" height="180"
    frameRate="25" sar="1:1" startWithSAP="3"
    bandwidth="43921">
  </Representation>
  <Representation
    id="h264ogop_mid" mimeType="video/mp4"
    codecs="avc1.64001e" width="640" height="360"
    frameRate="25" sar="1:1" startWithSAP="3"
    bandwidth="163912">
  </Representation>
     . . .
```

Segment Information

Segment Information proporciona información, principalmente, sobre los segmentos Initialization, Index y Bitstream Switching, sobre las URLs de los segmentos, sobre la disponibilidad de los segmentos en el caso de que MPD@type='dynamic'.

Initialization Segment information
La información sobre el Initialization Segment se proporciona mediante el elemento InitializationSegment, que puede aparecer en los elementos SegmentBase, SegmentList y SegmentTemplate o mediante el atributo @initialization del elemento SegmentTemplate. Si en una Representation no existe ningún elemento Initialization ni el atributo @initialization de SegmentTemplate, entonces todos los Media Segments de la Representation son auto-inicializables.

$**$< Identificador >$	**Parámetro de sustitución**
$$	Es una secuencia de escape. Se sustituye por un único "$"
$RepresentationID$	Se sustituye con el valor del atributo @id de la Representation en la que se encuentra
$Number$	Se sustituye con el número del Segment correspondiente
$Bandwidth$	Se sustituye con el valor del atributo @bandwidth de la Representation en la que se encuentra
$Time$	Se sustituye con el valor del atributo @t del elemento SegmentTimeline para el Segment al que se está accediendo. O bien se puede utilizar $Number$ o bien $Time$, pero no ambos al mismo tiempo.

Tabla 3.5: Identificadores para URL templates

Media Segment information

Cada Representation contiene una lista de Media Segments consecutivos, donde cada Media Segment tiene asignada una URL (y en ocasiones un rango de bytes), un número de segmento dentro de la Representation, un tiempo de inicio aproximado y una duración aproximada.
Estas características se especifican a través de los elementos SegmentList y SegmenTemplate. En el caso de que sea el elemento SegmentTemplate el que esté presente, la lista de segmentos se generará tal como se ha explicado en 3.1.4, y en el caso de que esten presenten uno o más elementos SegmentList, éstos contienen una lista de elementos SegmentURL, que representan las URLs y los rangos de bytes de los Media Segments. Finalmente en el caso de que no esté presente ni SegmentList ni SegmentTemplate, únicamente existirá un Media Segment, cuya URL se representa mediante en elemento BaseURL, y puede aparecer el elemento SegmentBase.

Index Segment information

Cada Segment normalmente tiene asignada Segment Index Information, que puede ser proporcionada explicitamente en un Index Segment, que puede ser representado por el elemento RepresentationIndex, por el atributo @index y el atributo @indexRange del elemento SegmentList.SegmentURL o por el atributo @index del elemento SegmentTemplate.

Bitstream Switching Segment information

Cada Representation tiene asignado como mucho un Bitstream Switching Segment, siempre en el caso que el valor del atributo @bitstreamSwitching sea 'true', y se representa mediante el elemento BitstreamSwitching o mediante el atributo @bitstreamSwitching del elemento SegmentTemplate

3.1.5. Media Content Component

Cada Adaptation Set contiene uno o más componentes de contenido multimedia, las propiedades de los cuales pueden ser descritas mediante el elemento ContentComponent o directamente en el elemento AdaptationSet en el caso de que sólo haya un componente de contenido multimedia en el Adaptation Set correspondiente. Tal como hemos visto en el apartado 3.1.2, el elemento ContentComponent se incluye dentro de un elemento AdaptationSet.
Algunos de los atributos que puede contener el elemento ContentComponent son: @id, @lang y @contentType entre otros.

3.1.6. Common attributes and elements

Los elementos AdaptationSet, Representation y SubRepresentation tienen algunos atributos y elementos comunes. Algunos de ellos son: @profiles, @width, @height, @frameRate, @mimeType y @codecs entre otros. De estos atributos que acabamos de nombrar, la mayoría son opcionales, excepto @mimeType y @codec, que como mínimo deben aparecer en el elemento Representation o bien en el elemento AdaptationSet.

3.1.7. Subset

Un Subset define un conjunto de uno o más Adaptation Sets, y se representa mediante el elemento Subset, dentro del elemento Period. El uso de este elemento permite restringir la combinación de Adaptation Sets a la hora de la presentación del contenido multimedia. De esta forma si existen uno o varios Subsets, el conjunto de Adaptation Set de los cuales se representa alguna de sus Representations debe ser un subconjunto de uno de estos Subsets.
Si un Adaptation Set no está explícitamente contenido en ningún Subset, está incluido implícitamente en todos los Subsets.
El elemento Subset únicamente contiene el atributo @contains, que contiene una lista de los valores de los atributos @id de los Adaptation Sets incluidos en el Subset.

3.2. Formatos de los segmentos

El formato de un segmento especifica la sintaxis y la semántica de los recursos asociados con HTTP-URLs en el MPD.
DASH se centra en formatos de segmentos basados en formatos de contenedores MPEG. En concreto estos contenedores son:

- ISO base media file format, definido en ISO/IEC 14496-12 (3.2.2)

- MPEG-2 Transport Stream, definido en ISO/IEC 13818-1 (3.2.3)

En primer lugar, en 3.2.1 veremos los tipos de segmentos que define DASH y a continuación describiremos brevemente los formatos que acabamos de nombrar.

3.2.1. Tipos de segmentos

DASH define cuatro tipo de segmentos:

- Initialization Segment
- Media Segment
- Index Segment
- Bitstream Switching Segment

Todos ellos son o pueden ser específicos del formato multimedia, por tanto en el apartado 3.2.2 se explicarán más detalles.

Initialization Segment

Contiene la información de inicialización necesaria para poder reproducir los segmentos de la Representation correspondiente.

Media Segment

Contiene y encapsula los streams multimedia.
Además, contiene un número de unidades de acceso completas, como mínimo debería contener un Stream Access Point (SAP) para cada stream multimedia que contenga y contiene más información necesaria para poder reproducir correctamente los streams multimedia.
Puede ser dividido en Subsegments mediante un Segment Index. En algunos formatos multimedia el Segment Index puede incluirse en el Media Segment, en cambio en otros debe aparecer en un Index Segment (3.2.1).

Index Segment

Contiene principalmente información sobre indexación para uno o más Media Segments.

Bitstream Switching Segment

Contiene la información necesaria para cambiar a la Representation a la que está asignado.

3.2.2. ISO base media file format

Define una estructura general para ficheros multimedia como audio y vídeo, especificado en ISO/IEC 14496-12 (MPEG-4 Part 12).
A continuación se ampliará la información de algunos de los tipos de segmentos nombrados en 3.2.1.

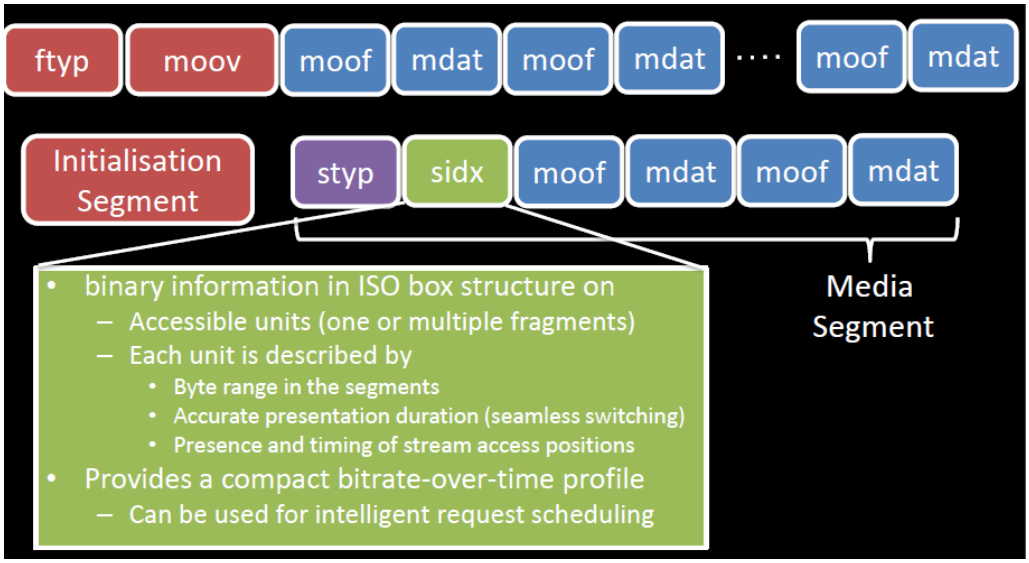


Figura 3.6: Formato de un segmento ISO base media file format en DASH[20].

Segment Index

Si existe este tipo de segmento, se utilizará el box 'sidx' definido en ISO/IEC 14496-12.

Initialization Segment

Contiene un 'ftyp' y un 'moov' box y no contiene ningún 'moof' box. Dentro del 'moov' box aparece el 'mvex' box, para indicar al cliente que debe esperar fragmentos de película.

Media Segments

Existen 3 tipos diferentes de Media Segments:

- Simple Media Segments
  - Puede tener un 'styp' box
  - Contiene uno o más conjuntos de fragmentos de película auto-contenidos, donde uno de estos conjuntos consta de un 'moof' box y un 'mdat' box, el cual contiene las muestras multimedia que no utilizan referencias de datos externos.
  - Contiene un 'traf' box, que a su vez contiene un 'tfdt' box.
  - Puede contener uno o más 'sidx' boxes, el primero de los cuales aparece antes que ningún 'moof' box.
- Indexed Media Segments
  - Sigue el formato de los segmentos Simple Media Segment y además en cada fragmento de película auto-contenido el box 'moof' es inmediatamente seguido por el box correspondiente 'mdat'.
  - El box 'sidx' es obligatorio, no opcional
- Sub-Indexed Media Segments
  - Sigue el formato de los segmentos Indexed Media Segments
  - Debe aparecer el box 'ssix', que sigue inmediatamente al box 'sidx'

3.2.3. MPEG-2 Transport Stream

Formato de contenedor que encapsula Packetized Elementary Streams (PES), con corrección de errores y con utilidades de sincronización de los streams para mantener la integridad de la transmisión cuando la señal se degrada. Está especificado y definido en ISO/IEC 13818-1.

En este proyecto nos hemos centrado en el formato ISO Base Media File Format, por tanto no describiremos este formato.

3.3. Profiles

Un profile impone un conjunto específico de restricciones, donde estas restricciones se suelen basar en características del MPD y en el formato de los segmentos, aunque también se pueden basar en otras características.
También se puede entender un profile como el permiso para un cliente DASH que únicamente implementa las características necesarias del perfil para procesar el MPD y los segmentos.

Cada profile tiene un identificador, que es una URI, y para indicar que un MPD cumple un profile determinado, se utiliza el atributo "@profiles" del elemento MPD. Como un MPD puede cumplir varios profiles, en este atributo podrán aparecer varios profiles separados mediante comas.

DASH define seis profiles, 3 basados en ISO Base Media File Format, dos basados en MPEG2-TS y el último que los engloba a todos. En la figura 3.7 podemos ver como se agrupan estos profiles.

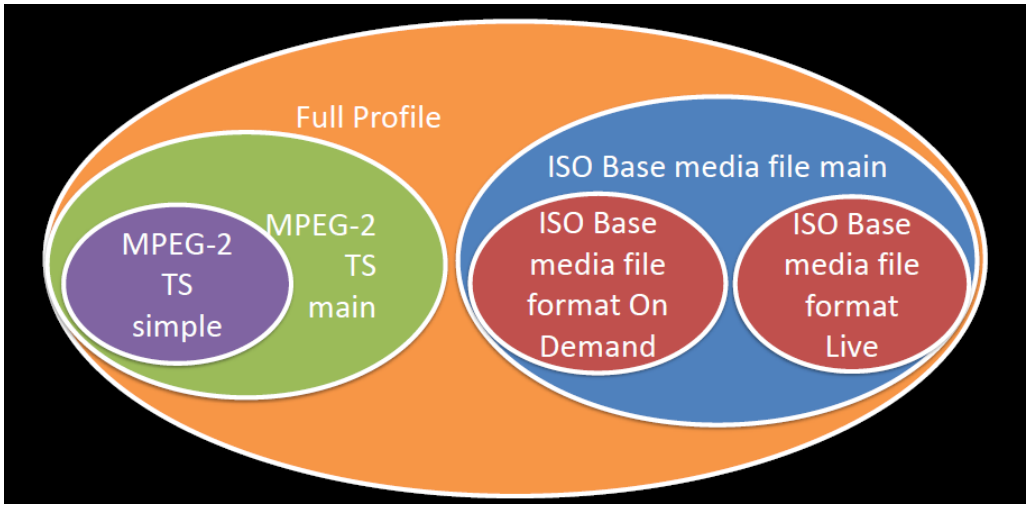


Figura 3.7: DASH profiles[20].

A continuación enumeraremos estos profiles con sus correspondientes identificadores:

- Full profile - "urn:mpeg:dash:profile:full:2011".
- ISO Base Media File Format on Demand - "urn:mpeg:dash:profile:isoff-on-demand:2011".
- ISO Base Media File Format Live - "urn:mpeg:dash:profile:isoff-live-2011".
- ISO Base Media File Format main - "urn:mpeg:dash:profile:isoff-main:2011".
- MPEG2-TS main - "urn:mpeg:dash:profile:mp2t-main:2011".
- MPEG2-TS simple - "urn:mpeg:dash:profile:mp2t-simple:2011".

Parte II

Desarrollo del proyecto

Capítulo 4

Especificación

La especificación describe el comportamiento esperado del sistema, es decir, describe las funciones que debe desempeñar el software y las exigencias que debe satisfacer.

En este capítulo definiremos los casos de uso del sistema, que describen la interacción entre el sistema y sus actores y posteriormente se describirá el modelo conceptual.

4.1. Casos de uso

En primer lugar veremos el diagrama de casos de uso (figura 4.1). Como podemos observar en esta figura, en el sistema existen dos tipos de actores, el usuario autenticado y el usuario no autenticado. Éste último únicamente podrá realizar Log In, ya que para el resto de casos de uso es necesario estar autenticado debido a que se debe acceder a la plataforma MIPAMS, tal como hemos comentado en el capítulo anterior.

A continuación detallaremos cada uno de los casos de uso que aparecen en la figura 4.1.

4.1.1. Log In

Actores: Usuario no autenticado.
Condición de activación: El usuario desea autenticarse en el sistema.
Precondiciones: -
Escenario principal:

1. El usuario introduce su nombre de usuario y su contraseña e indica al sistema que desea autenticarse.
2. El sistema se conecta con la plataforma MIPAMS, donde se comprobará que los datos son correctos.
3. En el caso de que los datos sean correctos, el usuario podrá acceder al panel principal del sistema.

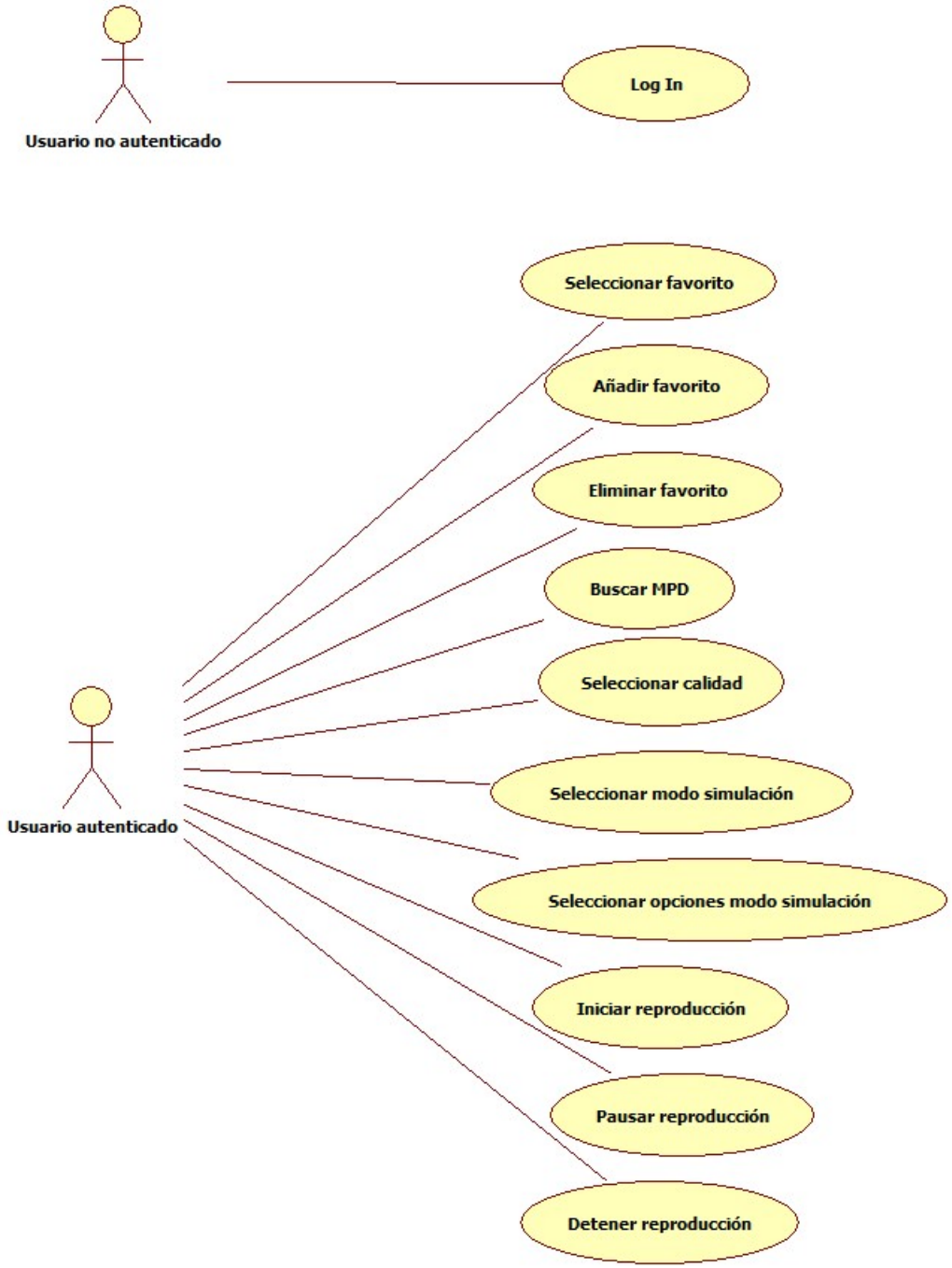


Figura 4.1: Diagrama de casos de uso

Escenario alternativo:

2. El sistema se conecta con la plataforma MIPAMS, donde se comprobará que los datos son correctos.
3. En el caso de que los datos no sean correctos el usuario no puede acceder al sistema, y éste le comunicará que el nombre o la contraseña son incorrectos.
4. El caso de uso vuelve al punto 1 del escenario principal.

4.1.2. Seleccionar favorito

Actores: Usuario autenticado.
Condición de activación: El usuario desea seleccionar un MPD de la lista de favoritos.
Precondiciones: -
Escenario principal:

1. El usuario indica al sistema que desea ver la lista de favoritos.
2. El sistema le muestra al usuario la lista de favoritos.
3. El usuario selecciona un elemento de la lista y le indica al sistema que desea continuar.
4. El sistema obtiene la URL del elemento seleccionado y la muestra en el panel principal.

4.1.3. Añadir favorito

Actores: Usuario autenticado.
Condición de activación: El usuario desea añadir un nuevo elemento a la lista de favoritos.
Precondiciones: -
Escenario principal:

1. El usuario indica al sistema que desea ver la lista de favoritos y posteriormente indica que desea añadir un nuevo elemento.
2. El sistema solicita al usuario los datos del nuevo elemento.
3. El usuario introduce el nombre y la URL del nuevo elemento e indica al sistema que añada dicho elemento a la lista de favoritos.
4. El sistema almacena el nuevo elemento y vuelve a mostrarle al usuario la lista de favoritos, que ya incluye el nuevo elemento.

Escenario alternativo:

3. El usuario introduce el nombre y la URL del nuevo elemento e indica al sistema que añada dicho elemento a la lista de favoritos.

4. Ya existe un elemento con ese nombre en la lista de favoritos. El sistema informa al usuario para que modifique el nombre.

5. El caso de uso vuelve al punto 2 del escenario principal.

4.1.4. Eliminar favorito

Actores: Usuario autenticado.
Condición de activación: El usuario desea eliminar un elemento de la lista de favoritos.
Precondiciones: -
Escenario principal:

1. El usuario indica al sistema que desea ver la lista de favoritos. Selecciona un elemento de la lista e indica al sistema que desea eliminarlo.

2. El sistema elimina el elemento de la lista y le muestra al usuario la lista actualizada.

4.1.5. Buscar MPD

Actores: Usuario autenticado
Condición de activación: El usuario desea obtener un MPD.
Precondiciones: El usuario previamente ha debido seleccionar un elemento de la lista de favoritos.
Escenario principal:

1. El usuario indica al sistema que desea obtener el MPD de la URL del elemento seleccionado previamente.

2. El sistema se conecta a la plataforma MIPAMS para comprobar si el usuario tiene permiso para acceder a ese MPD.

3. En caso de que tenga permiso, el sistema parsea el MPD y muestra al usuario la duración del contenido multimedia del MPD y una lista desplegable con las calidades disponibles. Además prepara todo lo necesario para una posible reproducción.

Escenario alternativo:

2. El sistema se conecta a la plataforma MIPAMS para comprobar si el usuario tiene permiso para acceder a ese MPD.

3. El usuario no tiene permiso. El sistema informa al usuario de que no tiene permiso para acceder a ese contenido.

4. El usuario deberá seleccionar otro elemento de la lista de favoritos.

5. El caso de uso vuelve al punto 1 del escenario principal.

4.1.6. Seleccionar calidad

Actores: Usuario autenticado.
Condición de activación: El usuario desea escoger una nueva calidad de entre las calidades disponibles.
Precondiciones: El usuario previamente ha debido obtener un MPD.
Escenario principal:

1. El usuario selecciona una nueva calidad de la lista de calidades cuando todavía no se ha iniciado la reproducción.

2. El sistema cambia la calidad seleccionada, realiza los cambios necesarios y queda preparado para una posible reproducción.

Escenario alternativo:

1. El usuario selecciona una nueva calidad de la lista de calidades a mitad de la reproducción.

2. El sistema pausa la reproducción, cambia la calidad seleccionada, realiza los cambios necesarios y continua con la reproducción.

4.1.7. Seleccionar modo simulación

Actores: Usuario autenticado.
Condición de activación: El usuario desea activar el modo simulación.
Precondiciones: El usuario previamente ha debido obtener un MPD. La reproducción no se debe haber iniciado.
Escenario principal:

1. El usuario indica al sistema que desea activar el modo simulación.

2. El sistema activa el modo simulación y realiza los cambios necesarios, dejando todo listo para una posible reproducción.

4.1.8. Seleccionar opciones modo simulación

Actores: Usuario autenticado.
Condición de activación: El usuario desea escoger las opciones del modo simulación.
Precondiciones: El usuario previamente debe de haber activado el modo simulación. La reproducción no se debe de haber iniciado.
Escenario principal:

1. El usuario indica al sistema que desea escoger las opciones del modo simulación.

2. El sistema le muestra al usuario las opciones que puede seleccionar.

3. El usuario selecciona las opciones que desea y se lo indica al sistema.

4. El sistema realiza los cambios necesarios y deja todo listo para una posible reproducción con el modo de simulación activado.

4.1.9. Iniciar reproducción

Actores: Usuario autenticado.
Condición de activación: El usuario desea reproducir el contenido multimedia del MPD seleccionado.
Precondiciones: El usuario previamente ha debido obtener un MPD.
Escenario principal:

1. El usuario indica al sistema que desea iniciar la reproducción.

2. El sistema obtiene un segmento del contenido multimedia y se lo envía a un reproductor multimedia.

3. Se repite el punto 2 hasta que se han descargado todos los segmentos del MPD o hasta que el usuario pausa o detiene la reproducción.

Escenario alternativo: La reproducción ya se había iniciado previamente.

1. El usuario indica al sistema que desea reanudar la reproducción.

2. El sistema indica al reproductor que reanude la reproducción y le envía los segmentos que se encuentran en el buffer (en el caso que haya alguno).

3. El caso de uso continúa en el punto 2.

4.1.10. Pausar reproducción

Actores: Usuario autenticado.
Condición de activación: El usuario desea pausar la reproducción.
Precondiciones: La reproducción se debe haber iniciado previamente.
Escenario principal:

1. El usuario indica al sistema que desea pausar la reproducción.
2. El sistema indica al reproductor que pause la reproducción.
3. El sistema continua continúa obteniendo segmentos y los va almacenando en un buffer.

4.1.11. Detener reproducción

Actores: Usuario autenticado.
Condición de activación: El usuario desea detener la reproducción.
Precondiciones: La reproducción se debe haber iniciado previamente.
Escenario principal:

1. El usuario indica al sistema que desea detener la reproducción.
2. El sistema indica al reproductor que detenga la reproducción.
3. El sistema no continua descargando segmentos y elimina los ya descargados. Además se prepara lo necesario por si se vuelve a iniciar la repoducción.

4.2. Modelo conceptual

El modelo conceptual representa los conceptos que existen en el sistema y las relaciones entre ellos.
A continuación mostraremos el diagrama de clases (figura 4.2), el cual representa los conceptos mediante clases y sus correspondiente atributos, y las relaciones mediante las asociaciones entre clases.

La clase DASH Panel es el "centro" del sistema y se encarga de preparar todo lo necesario para que la clase Segment Downloader vaya descargando los segmentos y se los vaya enviando a Player. La clase MPD Parser se encarga de parsear el MPD y obtener los datos necesarios para poder reproducir el contenido multimedia.

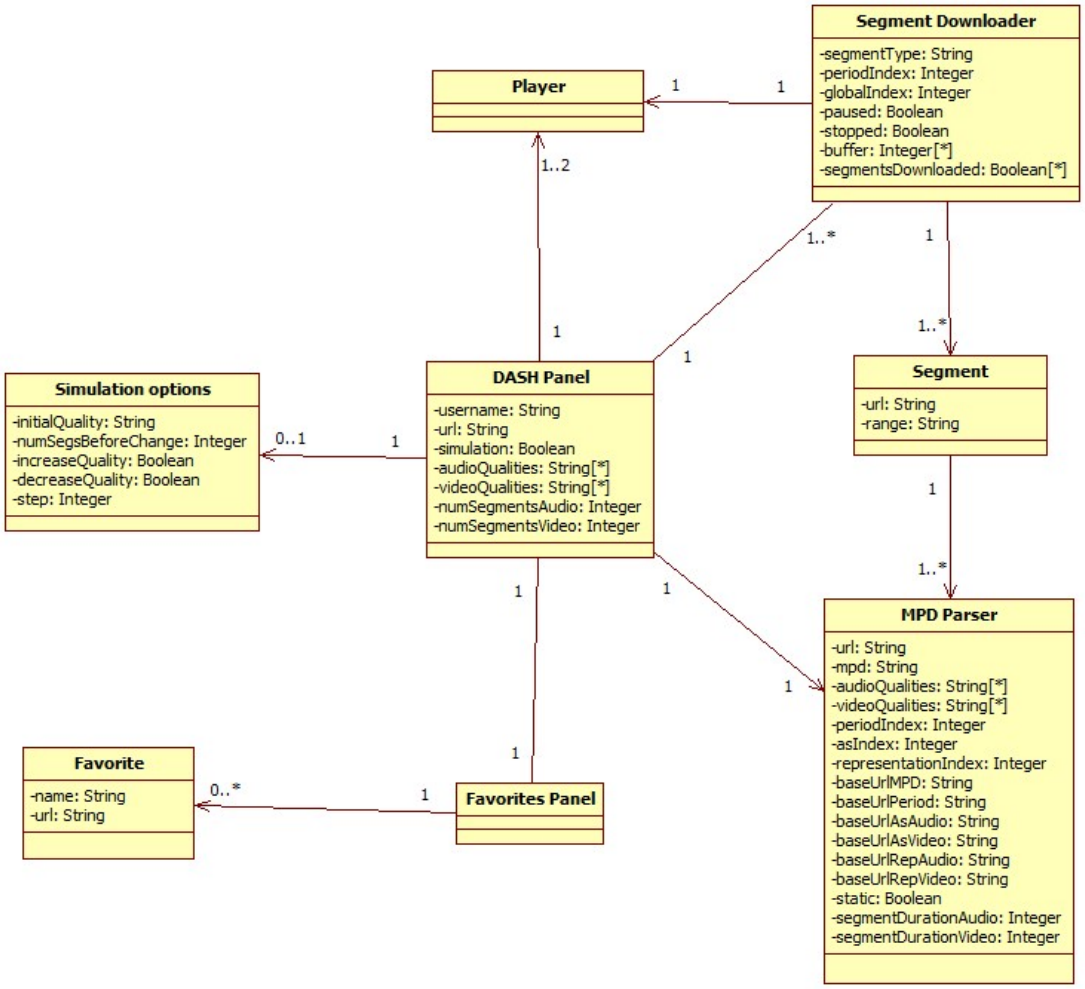


Figura 4.2: Diagrama de clases

Capítulo 5

Diseño

El diseño describe el proceso de definición de la arquitectura, componentes y otras características del sistema, teniendo en cuenta lo que se ha visto en la fase de especificación.

En este capítulo veremos algunos diagramas de secuencia, que se encargan de detallar el funcionamiento del sistema para los casos de uso vistos en la fase de especificación.

5.1. Diagramas de secuencia

En este apartado veremos los diagramas de secuencia de algunos de los casos de uso más importantes del sistema.

5.1.1. Buscar MPD

En la figura 5.1 podemos ver el diagrama de secuencia del caso de uso Buscar MPD. En él podemos observar como en primer lugar se accede a la plataforma MIPAMS para comprobar si el usuario tiene permiso para obtener el MPD que previamente ha seleccionado de la lista de favoritos. En el caso de que el usuario tenga permiso, se parsea el MPD, en la función parse(url). En esta función se setean todos los atributos de la instancia de MPD Parser. Después la instancia de DASH Panel obtiene los atributos necesarios y crea una o dos instancias de Segment Downloader, en función de si el MPD contiene uno o dos streams multimedia (vídeo y audio). Finalmente la instancia de DASH Panel le pasa a las instancias de Segment Downloader las urls (con su range, en caso de que exista) de los segmentos del primer Period del MPD. De esta forma las instancias de Segment Dowloader ya están preparadas para cuando se inicie la reproducción.

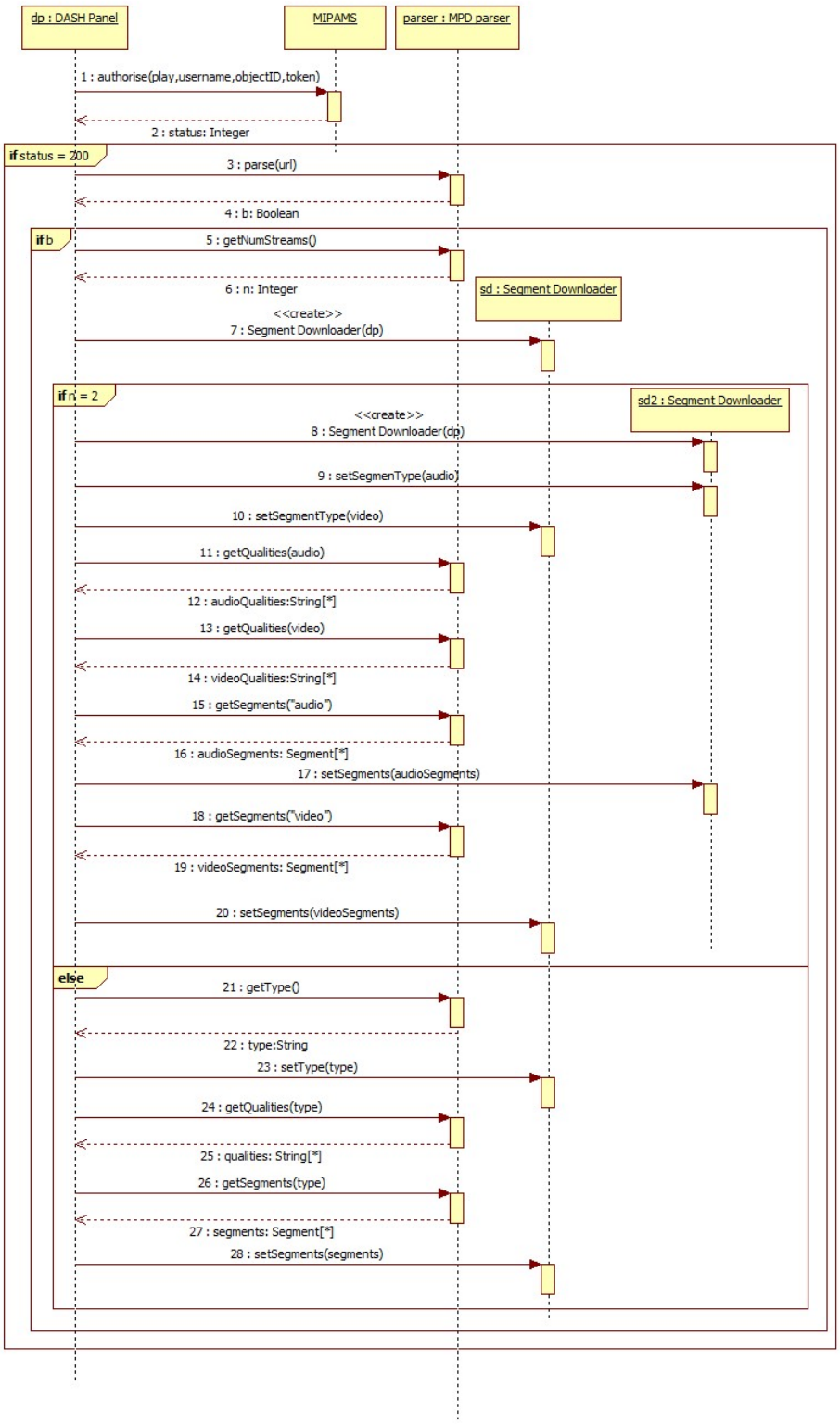


Figura 5.1: Diagrama de secuencia: Buscar MPD

5.1.2. Seleccionar calidad

En la figura 5.2 podemos ver el diagrama de secuencia del caso de uso Seleccionar calidad. En él podemos ver como la instancia de DASH Panel llama a la función changeQuality(...) de MPD Parser, pasándole la nueva calidad seleccionada y sobre que stream se ha cambiado la calidad (audio o vídeo). En esta función se cambia el atributo representationIndex y se obtienen las urls de la nueva calidad, debido a que un cambio de calidad implica que se cambia de Representation del MPD. Finalmente se le pasan las nuevas urls a la instancia de Segment Downloader correspondiente.

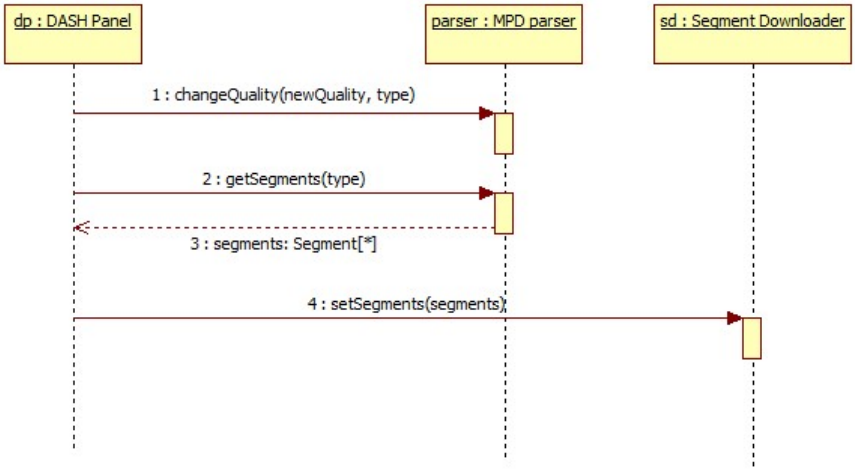


Figura 5.2: Diagrama de secuencia: Seleccionar calidad

5.1.3. Iniciar reproducción

En la figura 5.3 podemos ver el diagrama de secuencia del caso de uso Iniciar reproducción. Este diagrama representa el caso en el que únicamente hay un tipo de stream en el MPD, y por eso sólo aparece una instancia de Segment Downloader. En el caso de que hubiese dos streams, habría que realizar los mismos pasos con la segunda instancia que con la primera.
La instancia de Segment Downloader se encarga de crear la instancia de Player (en el caso que no exista ninguna) y después se la pasa a DASH Panel, de forma que si se desea reproducir otro MPD, se puede reaprovechar la misma instancia de Player.
A continuación se inicializan algunos atributos de la instancia de Segment Downloader y posteriormente se le indica que puede iniciar la reproducción, llamando a la función startStreaming(...). En esta función se irán descargando los segmentos y pasándoselos al Player o almacenándolos en un buffer en el caso de que la reproducción haya sido pausada.

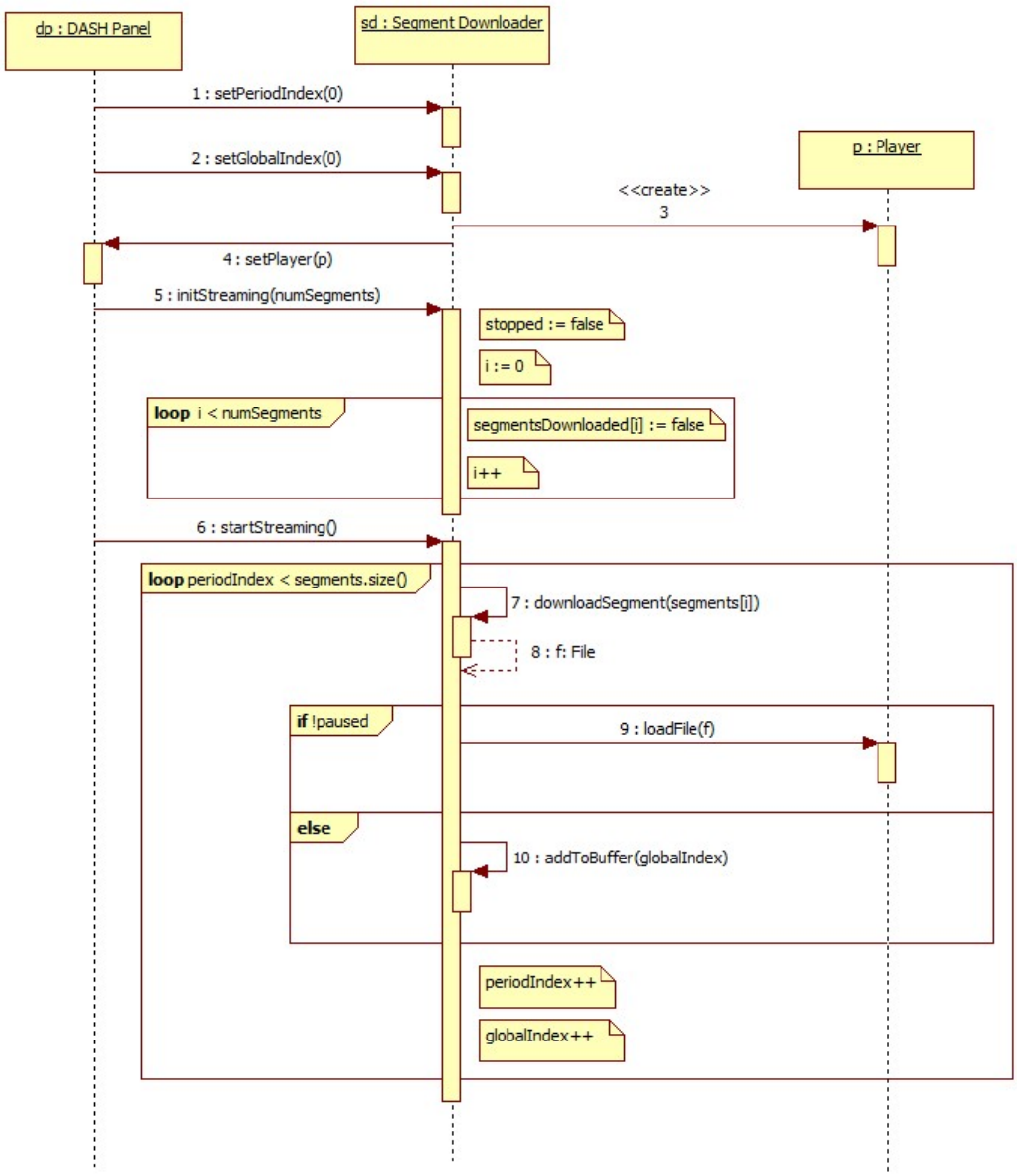


Figura 5.3: Diagrama de secuencia: Iniciar reproducción

Capítulo 6

Implementación

En este capítulo explicaremos algunos detalles de la implementación de la aplicación y los problemas que han ido apareciendo. También hablaremos sobre el reproductor utilizado.

Para implementar la aplicación se ha utilizado el lenguaje de programación Java debido a que es un lenguaje multiplataforma y también debido a que en un futuro se pueda portar a Android de una forma sencilla.

6.1. Reproductor

A la hora de buscar un reproductor para utilizar en la aplicación el primero en el que se pensó fue en VLC Media Player debido a que soporta una gran variedad de códecs, es open-source y además es utilizado por un grupo de la universidad de Klagenfurt que está trabajando en DASH [15].
Lo primero que se hizo fue descargar la última versión de VLC, la cual contenía un plugin de DASH y realizar pruebas con los MPDs del dataset del grupo de Klagenfurt[4][6]. Estas pruebas funcionaron correctamente, así que después se intentó reproducir un segmento de un MPD , concatenado con su segmento de inicialización, pero no se reproducía nada. Por tanto se decidió estudiar el código de VLC y debugar cuando se reproducía un MPD para ver que tratamiento realizaba VLC con los segmentos para poder reproducirlos. Pero el código era demasiado extenso y era muy difícil encontrar una solución analizándolo o debugándolo, así que se decidió continuar investigando para encontrar otro reproductor compatible.
Finalmente se encontró un trabajo[3] sobre DASH y en él se utiliza MPlayer[1] para reproducir los segmentos de los MPDs. Por tanto se probó a reproducir un segmento concatenado con su correspondiente segmento de inicialización y funcionó correctamente de forma que se decidió utilizar este reproductor para la aplicación.
Para utilizar MPlayer desde la aplicación, se crea un proceso que se encarga de llamar al

[1]MPlayer: reproductor multimedia, multiplataforma, que es capaz de reproducir una gran variedad de formatos.

Figura 6.1: Logo de MPlayer

reproductor. Dado que queremos que la aplicación se pueda ejecutar tanto en Windows como en Linux, a la hora de crear este proceso deberemos tener en cuenta el sistema operativo sobre el que se está ejecutando. A continuación podemos ver el fragmento de código donde se crea este proceso.

```
Process mplayerProcess = null;

...

String OS = System.getProperty("os.name");
String OS_Name = OS.split(" ")[0];
if (OS_Name.equals("Windows")){
  mplayerProcess = Runtime.getRuntime()
    .exec("smplayer-portable-0.8.0/mplayer/mplayer.exe
       -fixed-vo -vo gl2 -slave -xy 600 -idle -quiet ");
  jframe.setMplayerProcess(mplayerProcess, segmentType);
}
else if (OS_Name.equals("Linux")){
  mplayerProcess = Runtime.getRuntime()
    .exec("mplayer_linux/mplayer
      -fixed-vo -slave -idle -quiet -vo gl2 -xy 600 ");
  jframe.setMplayerProcess(mplayerProcess, segmentType);
}
```

Como podemos ver, en la llamada a MPlayer le pasamos diversas opciones. A continuación explicaremos para que sirve cada una.

- **-fixed-vo:** Fuerza un sistema de vídeo fijo para múltiples archivos, por lo tanto sólo se abrirá una ventana para todos ellos.
- **-vo:** Sirve para indicar el controlador de salida de vídeo
- **-slave:** Modo esclavo, en el cual MPlayer trabaja como backend para otros programas. En lugar de capturar eventos de teclado, leerá comandos de stdin separados por "\n".
- **-quiet:** Hace que MPlayer muestre menos información.

- **-idle:** Hace que MPlayer espere en modo ocioso en lugar de salir cuando no hay un fichero para reproducir.

- **-xy:** Si el valor es <=8, escala la imagen en un factor igual al valor. En el caso de que el valor sea >8, establece el ancho al valor y calcula el alto para mantener la relación de aspecto correcta.

Como hemos utilizado la opción -slave, MPlayer aceptará comandos de stdin, tal como acabamos de comentar. De esta forma podemos enviar ficheros para que los vaya reproduciendo, podemos indicar que pause la reproducción, que la pare, podemos obtener el nombre del fichero que se está reproduciendo, etc.
Para poder obtener información de MPlayer tenemos que poder leer de stdout y de stderr, y para ello hemos creado una clase auxiliar llamada LineRedirecter, que es un thread que se encarga de leer de un input stream, línea a línea, y lo escribe en un output stream. A continuación podemos ver el código de dicha clase.

```
class LineRedirecter extends Thread {
  /** The input stream to read from. */
  private InputStream in;
  /** The output stream to write to. */
  private OutputStream out;

  /**
   * @param in the input stream to read from.
   * @param out the output stream to write to.
   * @param prefix the prefix used to prefix the
            lines when outputting to the logger.
   */
  LineRedirecter(InputStream in, OutputStream out) {
    this.in = in;
    this.out = out;
  }
  public void run(){
    try {
      // creates the decorating reader and writer
      BufferedReader reader =
             new BufferedReader(new InputStreamReader(in));
      PrintStream printStream = new PrintStream(out);
      String line;
```

```
      // read line by line
      while ( (line = reader.readLine()) != null) {
            printStream.println(line);
      }
    }catch (IOException ioe) {
       ioe.printStackTrace();
    }
  }
}
```

Finalmente, con el siguiente código ya tenemos todo listo para poder enviar comandos a MPlayer y leer de su salida, donde utilizaremos *mplayerIn* para enviar comandos y *mplayerOutErr* para leer de la salida.

```
/* create the piped streams where to redirect the standard
   output and error of MPlayer*/
  PipedInputStream  readFrom = new PipedInputStream(1024*1024);
  PipedOutputStream writeTo = new PipedOutputStream(readFrom);
  mplayerOutErr =
        new BufferedReader(new InputStreamReader(readFrom));
  jframe.setMplayerOutErr(mplayerOutErr,segmentType);

/* create the threads to redirect the standard output
        and error of MPlayer */
  new LineRedirecter(mplayerProcess.getInputStream(),
    writeTo).start();
  new LineRedirecter(mplayerProcess.getErrorStream(),
    writeTo).start();

// the standard input of MPlayer
  mplayerIn = new PrintStream(
  mplayerProcess.getOutputStream());
  jframe.setMplayerIn(mplayerIn, segmentType);
```

Ahora que ya podemos enviar comandos a MPlayer vamos a ver cuales de ellos se han utilizado en la aplicación:

- **loadfile <file> <append>:** Con este comando le podemos enviar ficheros a MPlayer para que los reproduzca. Si append es un valor diferente de cero, la reproducción en curso no se interrumpe y el fichero se encola, en cambio si es 0, se interrumpe la reproducción en curso y se empieza a reproducir el nuevo fichero.

- **pause:** Pausa o reanuda la reproducción, en función de en que estado se encontrase.

- **stop:** Detiene la reproducción.
- **quit:** Cierra MPlayer.
- **get_time_pos:** Muestra la posición actual, en segundos, del fichero que se está reproduciendo.
- **get_file_name**: Muestra el nombre del fichero que se está reproduciendo.

En el siguiente código podemos ver como enviar un comando a MPlayer.

```
mplayerIn.print("pause");
mplayerIn.print("\n");
mplayerIn.flush();
```

Pero algunos de los comandos que acabamos de ver los utilizamos para obtener una respuesta de MPlayer, y no únicamente para enviar una orden, como por ejemplo en el caso de pause. Para obtener esta respuesta, después de enviar el comando, debemos leer la salida de MPlayer (utilizando la variable "mplayerOutErr") hasta que encontremos la respuesta. A continuación podemos ver un ejemplo.

```
String answer;
String filename;

mplayerIn.print("get_file_name");
mplayerIn.print("\n");
mplayerIn.flush();

try {
  while ((answer = mplayerOutErr.readLine()) != null) {
    if (answer.startsWith("ANS_FILENAME=")) {
      filename = answer.substring("ANS_FILENAME=".length());
      break;
    }
  }

}
catch (IOException e) {
  e.printStackTrace();
}
```

6.2. Parsing del MPD

Tal como hemos comentado en capítulos anteriores, se debe parsear el MPD, que es un fichero XML, para obtener los segmentos de los streams multimedia y posteriormente reproducirlos.
Para realizar el parsing XML se investigó sobre las APIs existentes que se pudieran utilizar desde Java de una forma sencilla. Algunas de estas APIs son:

- **SAX (Simple API for XML):** Procesador de documentos de acceso en serie y basado en eventos. Procesa el documento de forma secuencial y a medida que va encontrando ciertas partes del XML (comienzo de una etiqueta, fin de una etiqueta) van saltando eventos que son recogidos por una clase (manejador de eventos) que actuará de una forma distinta en función del evento.
  Únicamente permite procesar documentos, pero no crearlos.

- **DOM (Document Object Model):** Procesa el documento XML al completo y genera un árbol de nodos relacionados entre ellos que representa a dicho documento. Es una especificación del W3C, de forma que su objetivo principal es proporcionar una interfaz que pueda ser utilizada en una gran variedad de entornos y lenguajes (Java, Javascript, C++, PHP ...).
  En este caso, además de procesar el documento, permite modificarlo y crear documentos XML nuevos.

- **JDOM:** API desarrollada y optimizada específicamente para Java que permite procesar ficheros XML, realizar búsquedas, modificaciones, generar nuevos ficheros y serializar ficheros. También genera un árbol de nodos, igual que en el caso de DOM, aunque presenta algunas diferencias como el nombre de algunos tipos de nodos, por ejemplo.

- **XOM (XML Object Model):** Del mismo modo que JDOM, XOM es una API desarrollada específicamente para Java que permite realizar las mismas operaciones que JDOM. Sus principios básicos son corrección, simplicidad y rendimiento.

Finalmente se decidió utilizar DOM (en concreto la implementación para Java, JAXP) , debido a que no es una API específica de ningún lenguaje y de esta forma si en un futuro se porta la aplicación a Android, será más sencillo realizar el parsing. Por otra parte también se podría haber escogido SAX, ya que tampoco es una API específica de ningún lenguaje, pero como la aplicación necesita acceder al MPD varias veces (cada vez que se cambia de calidad, cada vez que se cambia de Period ...) no basta con procesar el documento de forma secuencial, sino que se necesita tener el árbol de nodos que representa al documento.

A continuación describiremos brevemente los tipos de nodos de DOM que se han utilizado en la aplicación.

- **Document:** Es la raíz del árbol. Representa al documento entero.

- **Element:** Representa a una etiqueta. Puede contener texto, atributos y nodos hijos.
- **Attr:** Atributo de un Element.

A la hora de realizar el parsing del MPD, lo primero que hay que realizar es obtener el nodo Document a partir del fichero XML.

```
DocumentBuilder docBuilder = DocumentBuilderFactory
                             .newInstance()
                             .newDocumentBuilder();
Document doc = docBuilder.parse(url);
```

Una vez tenemos el nodo Document ya podemos recorrer el árbol de nodos y obtener la información que deseemos. También podemos obtener un conjunto de etiquetas a partir de su nombre, mediante la función *getElementsByTagName*, pero en el caso de que existan etiquetas con el mismo nombre en distintas partes del documento, puede que esta función no sea del todo útil. Veamos un ejemplo de un MPD:

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                    ...
  <BaseURL>
    http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/...
  </BaseURL>
  <Period start="PT0S">
    <AdaptationSet bitstreamSwitching="true">
     <Representation id="0" ... bandwidth="45351">
      <BaseURL>bunny_15s_50kbit/</BaseURL>
      <SegmentBase>
        <Initialization sourceURL="bunny_50kbit_dash.mp4"/>
      </SegmentBase>
      <SegmentList duration="15">
        <SegmentURL media="bunny_15s1.m4s"/>
                    ...
      </SegmentList>
     </Representation>
    </AdaptationSet>
  </Period>
</MPD>
```

Como podemos ver existen dos etiquetas *BaseURL*, de forma que si deseamos obtener desde el nodo raíz su elemento *BaseURL*, al utilizar la función *getElementsByTagName* nos devolverá también la etiqueta *BaseURL* que se encuentra dentro de *Representation*. Por tanto para saber a qué elemento corresponde cada *BaseURL* habría que utilizar la función *getParentNode* en cada nodo *BaseURL*.
En este caso tampoco tiene mucha importancia, puesto que únicamente hay dos etiquetas *BaseURL* y no sería costoso utilizar la función *getParentNode*, pero en el caso en que el MPD fuese más grande y hubiese más etiquetas *BaseURL* sí que podría ser costoso.

Por este motivo y también por la facilidad a la hora de buscar un elemento dentro del árbol de nodos, se decidió utilizar también, para algunas tareas, XPath (XML Path Language), que es un lenguaje que permite construir expresiones para recorrer y procesar un documento XML. Básicamente se ha utilizado este lenguaje para evitar tener que recorrer todo el árbol siempre, y poder encontrar un elemento mediante su path, aunque también se han utilizado las funciones *getElementsByTagName* y *getChildNodes*.

Para ello se ha creado una función que dado un path devuelve una lista de nodos del elemento que se encuentra en *path*. A continuación podemos ver el código.

```
private NodeList getNodeList(String path) {
  XPathExpression xpe;
  NodeList list = null;
  try {
    xpe = XPathFactory.newInstance().newXPath().compile(path);
    list = (NodeList) xpe.evaluate(this.document,
                                   XPathConstants.NODESET);
  }
  catch (XPathExpressionException e) {
    e.printStackTrace();
  }
  return list;
}
```

Pero en ocasiones no nos interesa obtener una lista de nodos, sino que nos interesa obtener uno en concreto, como por ejemplo cuando se cambia de Period y queremos obtener el nuevo para realizar todos los cambios necesarios y continuar descargando segmentos. A continuación podemos ver la función que se ha creado para obtener un elemento en concreto dado un path. Esta función utiliza la función que hemos visto anteriormente.

```
private Element getElement(String path, int item) {
  Element el = null;
  NodeList nl = this.getNodeList(path);
  if(nl != null) el = (Element) nl.item(item);
  return el;
}
```

6.3. Gestión de los segmentos

Después de hablar sobre el reproductor que utiliza la aplicación y sobre el parsing del MPD, en este apartado hablaremos de como se ha realizado la gestión de los segmentos, desde que se obtiene la URL correspondiente del MPD hasta que se le pasa el fichero al reproductor.
También explicaremos que procedimientos se han realizado cuando el usuario decide cambiar de posición en la reproducción.

6.3.1. Descarga de un segmento

Lo primero que vamos a ver es como obtener los segmentos a partir de su URL. Tal como hemos comentado en capítulos anteriores, los segmentos se obtienen mediante peticiones HTTP y para ello, en la aplicación se ha utilizado la clase HttpUrlConnection.
A partir de una URL, creamos una conexión HTTP, representada como una instancia de la clase HttpUrlConnection. Posteriormente proporcionamos la infomación necesaria a esta instancia, que en nuestro caso es el método de la petición (GET) y el rango de bytes en el caso de que exista uno. Después accedemos al contenido y lo descargamos con el método *getInputSTream*() y finalmente lo almacenamos en un fichero.
En el siguiente código podemos ver como se realiza este proceso.

```
String actualURL = urls.get(periodIndex).getUrl();
URL segment = new URL(actualURL);
HttpURLConnection connnection=(HttpURLConnection) segment.openConnection();
connnection.setRequestMethod("GET");

if (urls.get(periodIndex).getRange() != null){
  connnection.setRequestProperty("Range", urls.get(periodIndex).getRange());
}

is = connnection.getInputStream();
leido = is.read(array);

while (leido > 0 && !stopped) {
  fos.write(array, 0, leido);
  leido = is.read(array);
}
```

Pero antes de descargar un segmento, hay que crear el fichero en el que almacenaremos dicho segmento, concatenando siempre antes el segmento de inicialización. Lo que hacemos es crear un FileOutputStream y a continuación copiamos el fichero de inicialización (en el caso de que éste exista).
Una vez hecho esto, ya podemos almacenar el segmento descargado, tal como vemos en el bucle del código anterior.
En el siguiente código podemos ver como crear este fichero.

```
String destName = PATH+ segmentType +"_"+ FILENAME +"_"+ globalIndex +EXTENSION;
fos = new FileOutputStream(destName, true);

if(urls.get(0) != null && urls.get(periodIndex) != null) {
  copia = copia(PATH + segmentType +"_"+ FILENAME +"_init"+ EXTENSION, destName);
}
```

Finalmente, cuando ya hemos descargado el segmento y lo hemos almacenado en el fichero lo que queda por realizar es pasarselo al reproductor, tal como hemos visto en 6.1, o bien añadir el número de segmento a un array de enteros en el caso de que la reproducción esté pausada, de forma que cuando se reanude, se recorrerá este array y se le irán pasando los correspondiente ficheros al reproductor.
En el siguiente código podemos ver como se ha realizado esto último.

```
public void setPaused(boolean paused) {
  if(!paused && start) {
    for(int i = 0; i < segmentsToAdd.size(); i++){
      int s = segmentsToAdd.get(i);
      loadSegment(s,s);
    }
    segmentsToAdd.clear();
  }
  this.paused = paused;
}
```

6.3.2. Cambio de segmento

Durante la reproducción el usuario puede desear avanzar o retroceder a otro punto. Para permitir esto, la aplicación contiene un slider para que el usuario pueda desplazarse a otros puntos de la reproducción. Únicamente se permite cambiar al inicio de un segmento.
Cuando el usuario selecciona el slider y se desplaza a un punto determinado, se calcula

qué segmento corresponde a ese punto. A continuación se comprueba si el segmento ya ha sido descargado o no. En caso afirmativo, se le pasa al reproductor y se empieza a reproducir desde el inicio. En caso contrario, se descarga y cuando ya se ha descargado también se le pasa al reproductor. Después se continúan descargando los siguientes segmentos.
A la hora de cambiar de segmento, hay que comprobar si pertenece al Period actual o a otro, de modo que si no pertenece al actual, se obtienen las urls del nuevo Period, se descarga el segmento seleccionado, y se continúan descargando los segmentos del nuevo Period.

6.4. Lista de favoritos

La aplicación contiene una lista de favoritos, que no es más que una lista de pares de una URL de un MPD y un nombre que desee el usuario . En un principio se pensó en añadir esta lista por comodidad para el usuario, de forma que si desea reproducir el contenido del MPD, no tenga que buscar su URL en otro sitio y después introducirla en la aplicación. De esta forma el usuario podía tener sus MPDs "preferidos" a mano.
Pero posteriormente esta lista se convirtió en necesaria, debido a que para verificar que el usuario tiene permiso para poder acceder a un MPD se necesita un identificador, que en este caso es el nombre. De esta forma cuando el usuario almacene un MPD en su lista de favoritos, deberá poner el mismo nombre que tiene dicho MPD en la plataforma MIPAMS, para que la verificación de que tiene permiso se pueda realizar correctamente.

Para implementar esta lista se decidió utilizar SQLite[2], debido a que no necesita servidor y es fácil de utilizar. Para poder utilizar SQLite en Java es necesario utilizar el driver JDBC de SQLite.

Únicamente necesitamos una tabla con dos campos, *nombre* y *url*. En el siguiente código podemos ver como hemos realizado esto.

```
Class.forName("org.sqlite.JDBC");
Connection conn =
        DriverManager.getConnection("jdbc:sqlite:favorites.db");
Statement stat = conn.createStatement();
String sqlCreate = "CREATE TABLE IF NOT EXISTS Favorites
                  (name TEXT, url TEXT,PRIMARY KEY (name,url));";
stat.execute(sqlCreate);
```

Este código se ejecuta cada vez que el usuario desea ver la lista de favoritos, por eso se

[2]SQLite es un sistema de gestión de bases de datos relacional compatible con ACID(Atomicity, Consistency, Isolation and Durability) y contenido en una pequeña librería.

usa la sentencia "CREATE TABLE IF NOT EXISTS". De esta forma la primera vez que se accede a la lista se creará la tabla, y en caso de que accidentalmente se borre la base de datos también se volverá a crear.

Una vez creada la tabla ya podemos ejecutar las sentencias SQL que necesitemos, que en el caso de la aplicación son las siguientes:

```
SELECT * FROM Favorites;
INSERT INTO Favorites (name,url) VALUES ('name', 'URL');
DELETE FROM Favorites WHERE name='name' and url='URL';
```

Para ejecutar una sentencia SQL en SQLite se utiliza el método *executeQuery(query)*, que devuelve como resultado un ResultSet, que es un conjunto de tuplas o filas. En el siguiente código podemos ver un ejemplo.

```
ResultSet rs = stat.executeQuery("select * from Favorites;");
while (rs.next()) {
  FavoriteElement fe = new FavoriteElement(rs.getString(1), rs.getString(2));
  list.add(fe);
}
rs.close();

stat.close();
conn.close();
```

6.5. Modo simulación

Otra funcionalidad de la aplicación es el modo simulación, el cual simula cambios en el ancho de banda de forma que podamos observar si la calidad de los segmentos va cambiando.
Para implementar esto se ha utilizado un fichero *.properties* en el que se almacenan las características de la simulación que escoge el usuario y después se utilizan en la aplicación. En el caso de que el usuario no seleccione estas características y directamente inicie la reproducción con el modo simulación activado, se utilizaran unas características por defecto.
En primer lugar explicaremos qué es un fichero *.properties* y como trabajar con ellos en Java.
Un fichero *.properties* es un archivo que nos permite almacenar variables de configuración de nuestra aplicación, en concreto permite almacenar pares clave, valor. Veamos el fichero *.properties* que se ha utilizado en la aplicación.

```
#DASH simulation mode properties
empeora_calidad=0
mejora_calidad=2
num_segmentos=5
calidad_inicial=45351
```

Como podemos ver existen 4 características. Las dos primeras sirven para escoger entre mejorar la calidad de los segmentos o empeorarla y la cantidad de incremento o decremento. En el ejemplo, podemos ver que se ha decidido mejorar la calidad en 2. La siguiente característica indica cada cuantos segmentos se debe cambiar de calidad. En este caso cada vez que se han descargado 5 segmentos se cambiará la calidad. Finalmente, la última característica es la calidad con la cual se desea iniciar la reproducción.
Para trabajar con este tipo de ficheros se ha utilizado la clase java.util.Properties, mediante la cual cargamos el fichero. A continuación podemos ver un ejemplo de como leer de un fichero *.properties*.

```
Properties prop = new Properties();
prop.load(new FileInputStream("src/DASHApplication_pkg/DASH.properties"));

String s = prop.getProperty("num_segmentos");
String m = prop.getProperty("mejora_calidad");
String e = prop.getProperty("empeora_calidad");
```

Pero además de leer de este tipo de ficheros, también podemos modificarlos, como se puede observar en el siguiente código.

```
Properties prop = new Properties();
prop.setProperty("calidad_inicial",(String) initQ_comboBox.getSelectedItem());
prop.setProperty("num_segmentos",String.valueOf(segments_spinner.getValue()));

FileOutputStream out =
             new FileOutputStream("src/DASHApplication_pkg/DASH.properties");
prop.store(out, "DASH simulation mode properties");
out.close();
```

6.6. Cambio de calidad

Tal como ya hemos comentado en capítulos anteriores, DASH es un estándar para streaming adaptativo sobre HTTP y por tanto en la aplicación se debe gestionar el cambio de

calidad automático en función de los cambios en el ancho de banda y en la capacidad de CPU del cliente.
Cada vez que se descarga un segmento, se calcula cuanto tiempo se tarda desde que se empieza a descargar hasta que se ha almacenado en el fichero correspondiente y de esta forma calculamos el ancho de banda aproximado. En el siguiente código podemos ver como se realiza esto.

```
URL segment = new URL(actualURL);

HttpURLConnection c =(HttpURLConnection)segment.openConnection();
c.setRequestMethod("GET");
if (urls.get(periodIndex).getRange() != null){
  c.setRequestProperty("Range",urls.get(periodIndex).getRange());
}

ta = System.currentTimeMillis();

is = c.getInputStream();
length = c.getContentLength();

leido = is.read(array);
while (leido > 0 && !stopped) {
  fos.write(array, 0, leido);
  leido = is.read(array);
}

tb = System.currentTimeMillis();
b = this.bandwidth(ta, tb, length);

if (automatic && periodIndex != 0 && !simulation) {
  jframe.updateBandwidth(b, segmentType);
}
```

Después, en la función *updateBandwidth* se selecciona la calidad inferior más próxima al ancho de banda que se acaba de calcular, se obtienen las urls de la nueva Representation y se descarga el nuevo segmento de inicialización. Una vez descargado este segmento, se continúan descargando los segmentos de la nueva Representation.

Pero además de cambiar la calidad de forma automática, también puede que durante la reproducción el usuario desee cambiar manualmente a una cierta calidad, seleccionando una de las calidades de la lista. En este caso se obtiene el número de segmento que se está reproduciendo, se comprueba a qué Period pertenece, se obtienen las urls correspondientes y se descarga el mismo segmento que se estaba reproduciendo pero de la nueva calidad. Una vez descargado, el segmento se reproducirá desde el inicio, y a continuación

continuaran descargándose los segmentos posteriores.

6.7. Gestión de los streams

Todo lo que se ha explicado en los apartados anteriores, primero se implementó para MPDs que contenían un único stream multimedia, que podía ser tanto de audio como de vídeo, ya que prácticamente todos los ejemplos de MPD que se habían encontrado eran de un único stream.
Una vez que se finalizó la implementación para un único stream, se decidió ampliar la implementación para el caso de MPDs de dos streams, uno de audio y otro de vídeo. En la web de GPAC[3][10] se encontró un MPD que contenía un stream de audio y uno de vídeo, pero el audio finalizaba unos 30 segundos, aproximadamente, antes que el vídeo, y era difícil analizar si era problema de la aplicación o de la construcción del MPD, ya que el audio eran pitidos y el vídeo lo que se observa en la figura 6.2.

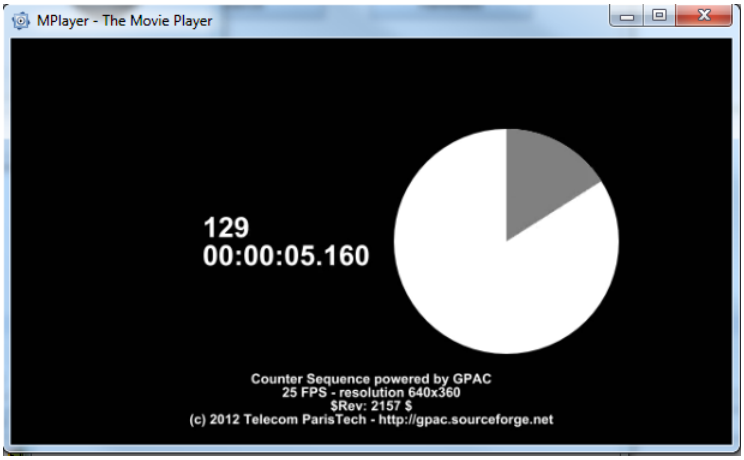


Figura 6.2: Ejemplo del vídeo de un MPD

Por tanto, se decidió utilizar un MPD del dataset del grupo de Klagenfurt[4][6], en concreto el de "Big Buck Bunny", obtener el audio del vídeo con el mismo nombre [2], extraer el audio con FFMpeg, segmentarlo con la herramienta MP4Box y añadir los segmentos de audio al MPD.
Al realizar las pruebas con este MPD se observó que la aplicación funcionaba bien, y que lo que ocurría en el caso anterior era debido a la construcción del MPD. El único problema que se encontró en la aplicación es que a medida que se avanza en la reproducción

[3]GPAC is an Open Source multimedia framework for research and academic purposes. The project covers different aspects of multimedia, with a focus on presentation technologies (graphics, animation and interactivity) and on multimedia packaging formats such as MP4.

se va produciendo un delay entre el audio y el vídeo. Esto ocurre en el caso de que la aplicación se ejecute en Windows, en cambio si se ejecuta en Linux lo que ocurre es que entre segmento y segmento en el caso de audio se para, aunque en el caso de vídeo se reproducen todos los segmentos de forma fluida. Esto no se ha podido solucionar y queda como trabajo futuro.

6.8. Interfaz gráfica

Después de ver algunos de los principales detalles de la implementación, vamos a ver la interfaz gráfica de la aplicación.
En primer lugar tenemos la ventana de login (figura 6.3), en la cual el usuario debe introducir su nombre de usuario y su password de MIPAMS. Mediante esta ventana podemos ejecutar el caso de uso Log In (4.1.1).

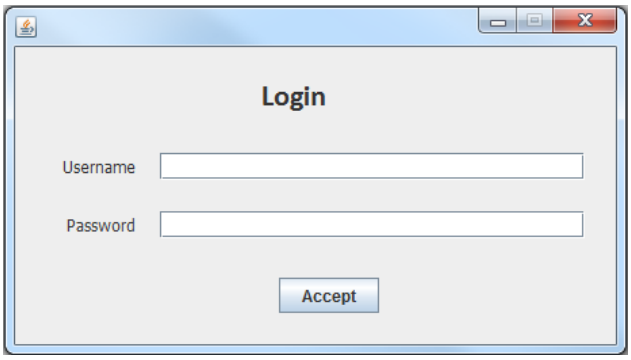


Figura 6.3: Ventana de login

Después de realizar el login, podremos acceder al panel principal (figura 6.4), desde el cual podemos acceder a la lista de favoritos y ejecutar el caso de uso Buscar MPD (4.1.5).

Si clicamos en Favorites se mostrará la ventana que contiene la lista de favoritos (figura 6.5). En esta ventana podemos ejecutar los casos de uso Seleccionar Favorito (4.1.2), Añadir Favorito (4.1.3) y Eliminar Favorito (4.1.4).

Para ejecutar los casos de uso Seleccionar Favorito o Eliminar Favorito tenemos que seleccionar un elemento de la lista y a continuación clicar en Select o Delete respectivamente. En el caso que deseemos ejecutar el caso de uso Añadir Favorito, tenemos que clicar en Add New y aparecerá ventana que se muestra en la figura 6.6.

En esta ventana debemos introducir el nombre y la URL del MPD que deseemos añadir.

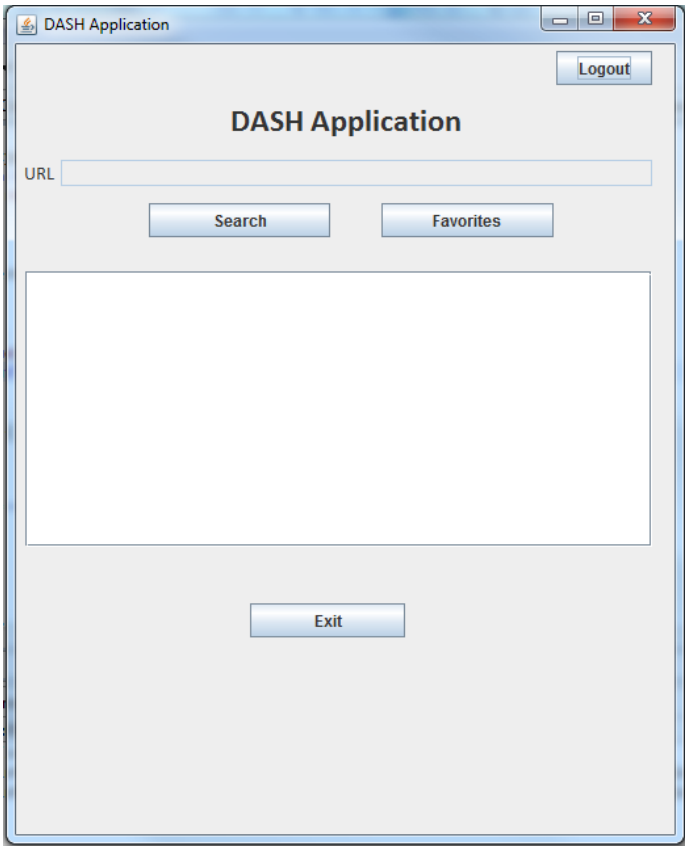


Figura 6.4: Panel principal

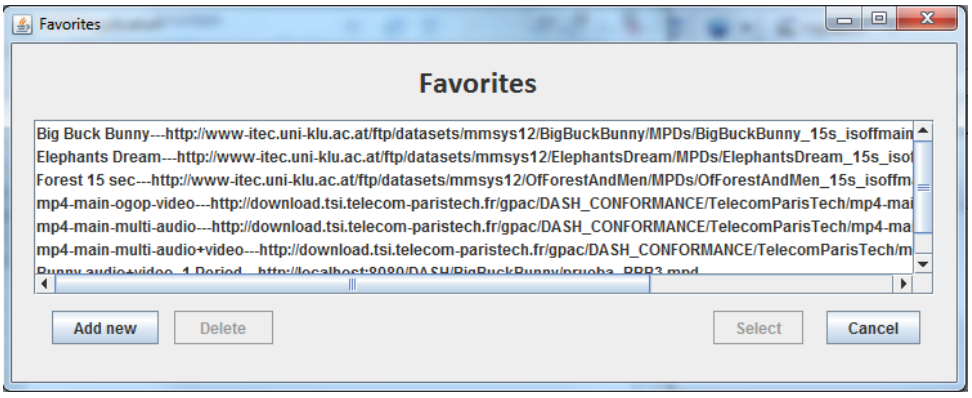


Figura 6.5: Ventana de favoritos

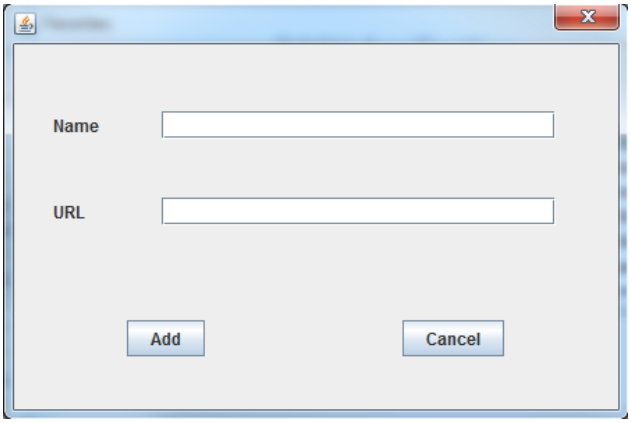


Figura 6.6: Ventana de añadir favorito

Una vez que cliquemos en Add, se volverá a mostrar la ventana que contiene la lista de favoritos que ahora también contendrá el nuevo elemento añadido.

Si ejecutamos el caso de uso Seleccionar Favorito, se volverá a mostrar el panel principal y a continuación podremos ejecutar el caso de uso Buscar MPD, clicando en Search. Una vez ejecutado este caso de uso, el panel principal mostrará más elementos, tal como podemos ver en la figura 6.7.

Desde esta ventana ahora podemos ejecutar los casos de uso Seleccionar calidad (4.1.6), Seleccionar modo simulación (4.1.7) e Iniciar Reproducción (4.1.9), además del caso de uso Buscar MPD, tal como hemos comentado antes.
Si deseamos ejecutar el caso de uso Seleccionar modo simulación, debemos marcar el checkbox simulation, y se mostrará un botón para poder ejecutar el caso de uso Seleccionar opciones modo simulación (4.1.8), tal como se muestra en la figura 6.8. También podemos observar como se desactiva la lista de calidades, debido a que en el modo simulación no podemos cambiar manualmente de calidad.
Si clicamos en este botón, se mostrará la ventana para seleccionar las opciones del modo simulación, tal como se muestra en la figura 6.9. En esta ventana aparecen las opciones que se explicaron en el apartado 6.5.

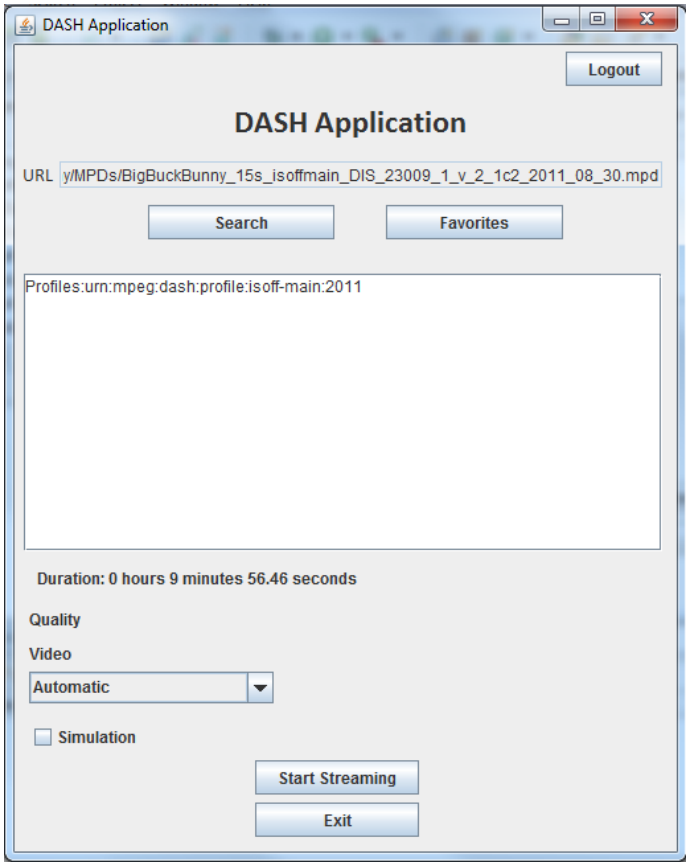


Figura 6.7: Panel principal después de ejecutar el caso de uso Buscar MPD

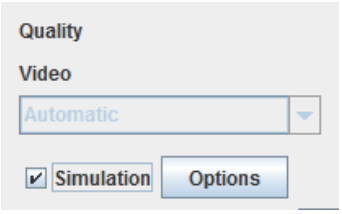


Figura 6.8: Parte del panel principal

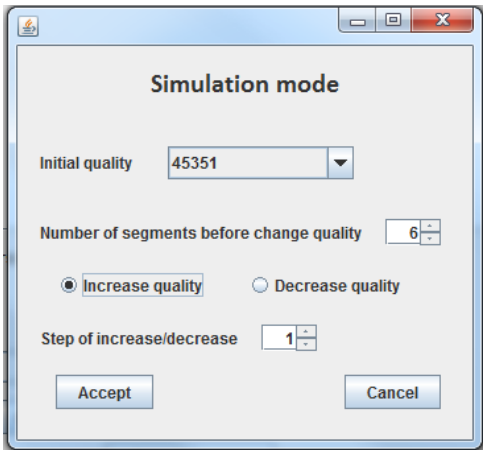


Figura 6.9: Ventana opciones modo simulacion

Para ejecutar el caso de uso Seleccionar calidad, únicamente debemos seleccionar un elemento de la lista desplegable, tal como podemos ver en la figura 6.10.

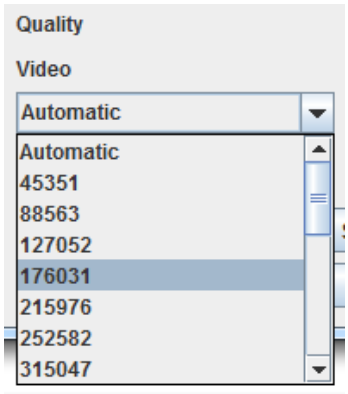


Figura 6.10: Lista desplegable de calidades

Una vez hemos seleccionado la calidad deseada o bien hemos seleccionado el modo simulación con sus correspondientes opciones, podemos ejecutar el caso de uso Iniciar reproducción, clicando en Start streaming. Al clicar este botón, aparecerán dos ventanas, la ventana de MPlayer (figura 6.11) donde podremos ver el vídeo (si el contenido multimedia es audio, esta ventana no aparecerá) y la ventana de los controles para la reproducción (figura 6.12).

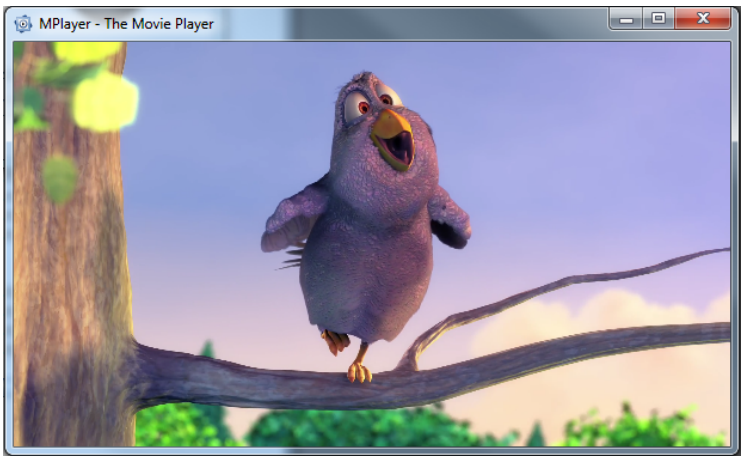


Figura 6.11: Ventana de MPlayer reproduciendo el vídeo de Big Buck Bunny

En la ventana de los controles podemos pausar, detener, iniciar o reanudar la reproducción. También podemos cambiar de posición en la reproducción, utilizando el slider, que va avanzando a medida que avanza la reproducción y también podemos ver un reloj que nos indica

el tiempo de reproducción y una barra de progreso que indica por donde se encuentra la descarga de los segmentos.

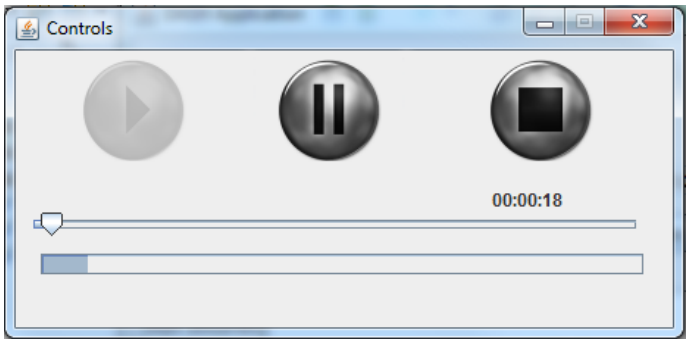


Figura 6.12: Ventana de los controles para la reproducción

Además a medida que avanza la reproducción, en el área de texto del panel principal podemos ir viendo las urls de los segmentos que han sido descargados, de forma que podemos observar los cambios de calidad que se van produciendo. En la figura 6.13 podemos ver un ejemplo.

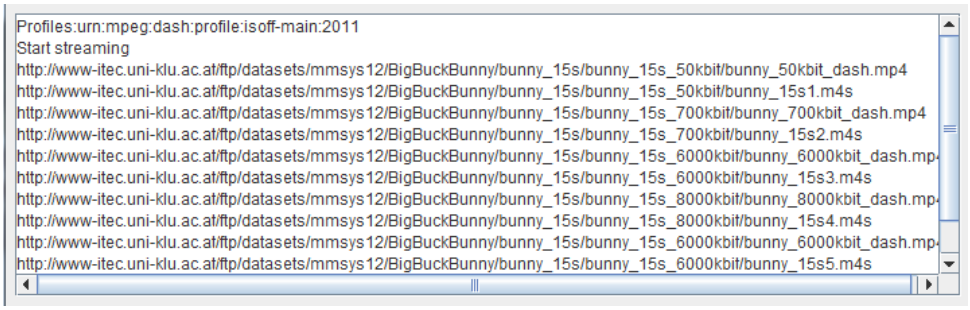


Figura 6.13: Área de texto del panel principal

Como podemos ver, la calidad de los segmentos va cambiando y cada vez que cambia se descarga el segmento de inicialización correspondiente y a continuación se descarga el segmento que toca.

Finalmente, cuando se produce algún error como por ejemplo que el usuario no está autorizado para acceder al MPD, se utiliza una ventana de alerta como la de la figura 6.14 para informar al usuario.

Figura 6.14: Ventana de alerta

Capítulo 7

Pruebas

En este capítulo explicaremos algunas de las pruebas que se han realizado para testear la aplicación y para estudiar el funcionamiento del estándar.

Tal como se ha comentado en apartados anteriores, la mayoría de los MPDs que se han utilizado para realizar pruebas han sido los del dataset del grupo de Klagenfurt y algunos de los de la web de GPAC. Sin embargo, todos estos MPDs tenían un único Period, de forma que se han modificado algunos de ellos repartiendo sus segmentos en más de un Period y así se ha podido comprobar si la aplicación funcionaba correctamente con este tipo de MPDs.
Además, tal como se comentó en el apartado 6.7, también se ha creado un MPD que contiene dos streams, uno de audio y uno de vídeo, para comprobar el funcionamiento de este tipo de MPDs.

7.1. Big Buck Bunny

La mayoría de las pruebas se han realizado con los MPDs de este vídeo de animación. En el dataset del grupo de Klagenfurt[4] podemos encontrar diferentes MPDs de este vídeo, que se diferencian por la duración de los segmentos. Las duraciones son las siguientes:

- 1 segundo
- 2 segundos
- 4 segundos
- 6 segundos
- 10 segundos

Figura 7.1: Big Buck Bunny[2]

- 15 segundos

En la tabla 7.1 podemos ver qué resolución se corresponde a cada bitrate en el caso del vídeo de Big Buck Bunny.

Bitrate	**Resolucion**
50 kbits/s	320x240
100 kbits/s	320x240
150 kbits/s	320x240
200 kbits/s	480x360
250 kbits/s	480x360
300 kbits/s	480x360
400 kbits/s	480x360
500 kbits/s	480x360
600 kbits/s	854x480
700 kbits/s	854x480
900 kbits/s	1280x720
1,2 Mbits/s	1280x720
1,5 Mbits/s	1280x720
2,0 Mbits/s	1280x720
2,5 Mbits/s	1920x1080
3,0 Mbits/s	1920x1080
4,0 Mbits/s	1920x1080
5,0 Mbits/s	1920x1080
6,0 Mbits/s	1920x1080
8,0 Mbits/s	1920x1080

Tabla 7.1: Bitrates y sus correspondientes resoluciones del vídeo Big Buck Bunny

Se han realizado 3 pruebas de cada uno de estos MPDs, excepto con el de los segmentos de 1 segundo, ya que la reproducción no era fluida y se paraba constantemente. A continuación mostraremos gráficos de la media de las 3 pruebas de cada MPD para poder ver como varía el bitrate de los segmentos durante la reproducción. En el eje horizontal se encuentran los números de los segmentos, y en el vertical el bitrate.

7.1.1. Segmentos de 2 segundos

En primer lugar podemos ver el gráfico (figura 7.2) de las pruebas con el MPD de segmentos de 2 segundos.

Como podemos observar, rápidamente se alcanza la calidad máxima, pero se producen bastantes cambios de calidad. A pesar de todo estos cambios no son demasiado bruscos, y durante toda la reproducción se mantiene una calidad bastante aceptable. Un aspecto

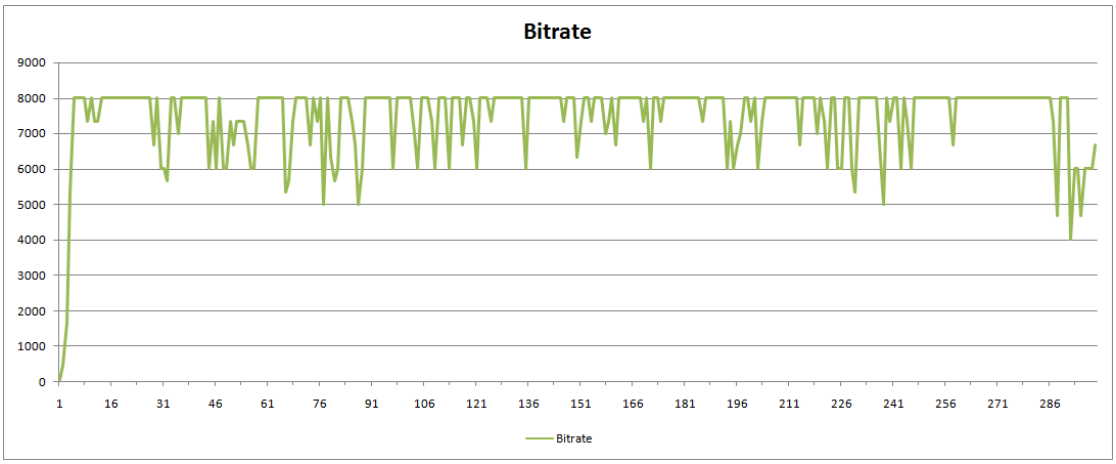


Figura 7.2: Bitrate medio de las pruebas con segmentos de 2 segundos del vídeo Big Buck Bunny

a destacar es que durante el primer minuto, la reproducción se paró unas diez veces, alrededor del segundo 10, del segundo 30 y del segundo 50. Una vez superado el minuto, el buffer de segmentos iba aumentando y no se volvió a parar.
En las tres pruebas se produjeron alrededor de 80 cambios de calidad, la mayoría de ellos alternando entre los bitrates 8 Mbits/s y 6 Mbits/s. Estos cambios no se produjeron en los mismos segmentos en las tres pruebas, aunque en algunos casos sí. Por ejemplo entre los segmentos 44-50 se produjeron los mismos cambios en la segunda y en la tercera prueba, y entre los segmentos 223-229 se produjeron los mismos cambios en las tres pruebas.

7.1.2. Segmentos de 4 segundos

En este caso (figura 7.3) también se alcanza rápidamente la calidad máxima, pero en este caso se mantiene durante más tiempo, de forma que no se producen tantos cambios de calidad como en el caso anterior. La reproducción también se detuvo en estas pruebas durante el primer minuto, pero únicamente unas 3 veces, alrededor de los segundos 30 y 50.
En las tres pruebas se produjeron alrededor de 10 cambios de calidad, alternando entre los bitrates 8 Mbits/s, 6 Mbits/s y 5 Mbits/s en la mayoría de los casos. En este caso no se produjo un cambio igual en las tres pruebas, aunque sí parecido, por ejemplo en las tres pruebas se produjo un cambio a 5 Mbits/s pero en un caso en el segmento 26, en otro en el 23 y en otro en el 22.

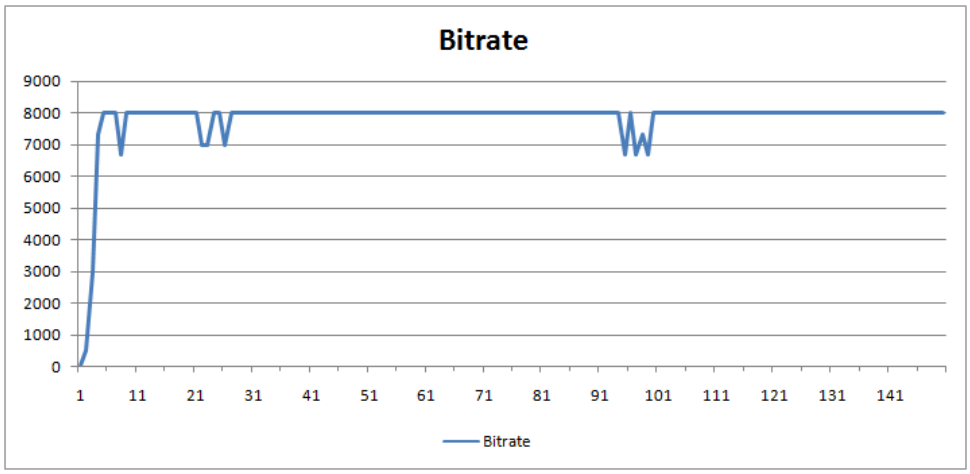


Figura 7.3: Bitrate medio de las pruebas con segmentos de 4 segundos del vídeo Big Buck Bunny

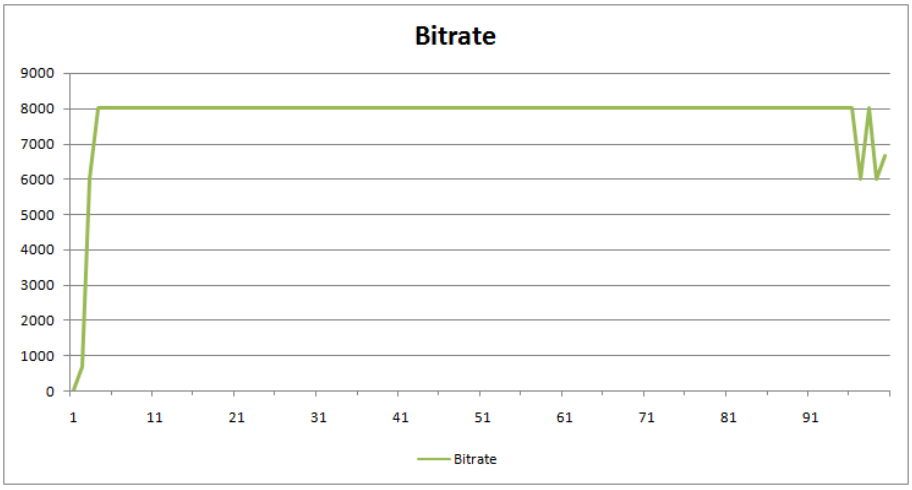


Figura 7.4: Bitrate medio de las pruebas con segmentos de 6 segundos del vídeo Big Buck Bunny

7.1.3. Segmentos de 6 segundos

En este gráfico (figura 7.4) podemos ver que prácticamente durante toda la reproducción se mantiene la calidad máxima, excepto en los últimos segmentos que se produce alguna pequeña variación. En este caso la reproducción se detuvo un par de veces durante el primer minuto.
En las tres pruebas prácticamente se produjeron los mismos cambios. Los cambios iniciales hasta que se alcanza la calidad máxima fueron los mismos en las tres pruebas y después en los segmentos 97 y 98 también se produjeron los mismos cambios. Finalmente en los últimos segmentos de la tercera prueba se produjeron un par de cambios que en las otras dos no.

7.1.4. Segmentos de 10 segundos

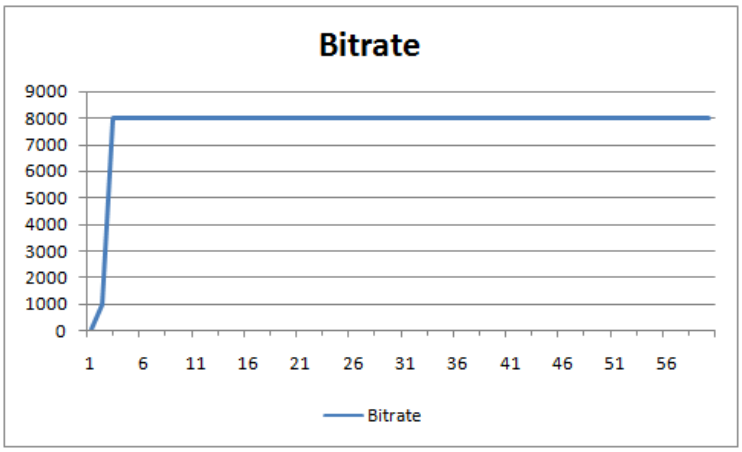


Figura 7.5: Bitrate medio de las pruebas con segmentos de 10 segundos del vídeo Big Buck Bunny

En estas pruebas (figura 7.5) cuando se alcanza la calidad máxima se mantiene durante toda la reproducción y no se produce ningún cambio. Además la reproducción no se ha detenido ninguna vez.
En las tres pruebas únicamente se produjeron cambios de calidad al inicio de la reproducción, hasta que se alcanzó la calidad máxima. Estos cambios fueron exactamente los mismos en la primera y la tercera prueba y en la segunda se produjo un cambio diferente en el segundo segmento.

7.1.5. Segmentos de 15 segundos

Finalmente veremos el gráfico de las pruebas realizadas con el MPD de segmentos de 15 segundos.

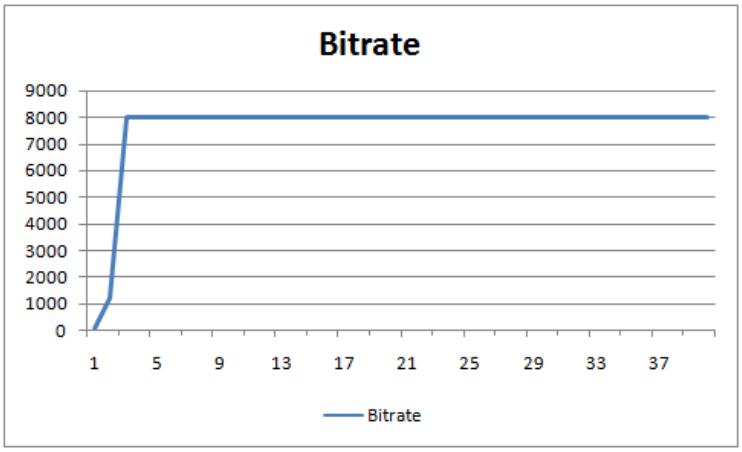


Figura 7.6: Bitrate medio de las pruebas con segmentos de 15 segundos del vídeo Big Buck Bunny

En este último gráfico figura 7.6 podemos observar que los resultados obtenidos en las pruebas con los segmentos de 15 segundos son prácticamente iguales que la de los segmentos de 10 segundos. En este caso tampoco se ha detenido la reproducción en ningún momento.
Finalmente en este caso se produjeron los mismos cambios en las tres pruebas, y ocurrió lo mismo que en el caso de los segmentos de 10 segundos, únicamente hubo cambios hasta que se alcanzó la calidad máxima.

7.1.6. Conclusiones

Una vez realizadas todas estas pruebas y viendo sus correspondientes gráficos, podríamos concluir que a mayor duración de los segmentos, menos cambios de calidad se producen y menos veces se detiene la reproducción y que los cambios no siempre se producen en los mismos segmentos aunque en algunas ocasiones sí.

7.2. Of Forest and Men

Después de haber realizado pruebas con el vídeo de animación Big Buck Bunny, se han realizado pruebas con la pequeña película Of Forest and Men, realizada para el Año Internacional de los Bosques. Se han realizado las mismas pruebas que en el caso anterior, 3 repeticiones con cada MPD, con los MPDs de 2, 4, 6, 10 y 15 segundos. En primer lugar veremos el gráfico de las pruebas con el MPD de segmentos de 2 segundos.

En la tabla 7.2 podemos ver qué resolución se corresponde a cada bitrate en el caso del vídeo de Of Forest and Men.

Bitrate	Resolucion
50 kbits/s	320x240
100 kbits/s	320x240
150 kbits/s	320x240
200 kbits/s	480x360
250 kbits/s	480x360
300 kbits/s	480x360
400 kbits/s	480x360
500 kbits/s	854x480
600 kbits/s	854x480
700 kbits/s	854x480
900 kbits/s	1280x720
1,2 Mbits/s	1280x720
1,5 Mbits/s	1280x720
2,0 Mbits/s	1920x1080
2,5 Mbits/s	1920x1080
3,0 Mbits/s	1920x1080
4,0 Mbits/s	1920x1080
5,0 Mbits/s	1920x1080
6,0 Mbits/s	1920x1080

Tabla 7.2: Bitrates y sus correspondientes resoluciones del vídeo Of Forest and Men

7.2.1. Segmentos de 2 segundos

Como podemos ver en la figura 7.7, se producen bastantes cambios de calidad, algunos de ellos bastante significativos, como es el caso de los últimos segmentos, donde la calidad llega a bajar casi hasta la calidad inicial. En las dos primeras pruebas la reproducción se detenía cada 2-6 segundos desde el minuto 4:24 hasta el minuto 5:15, en cambio en la tercera reproducción no se detuvo ninguna vez.
En las tres pruebas se produjeron alrededor de 54 cambios de calidad, la mayoría de ellos alternando entre los bitrates 5 Mbits/s, 4 Mbits/s y 3 Mbits/s. Estos cambios no se produjeron en los mismos segmentos en las tres pruebas, aunque en algunos casos sí. Por ejemplo en las tres pruebas en los últimos ocho segmentos se produjeron prácticamente los mismos cambios de calidad.

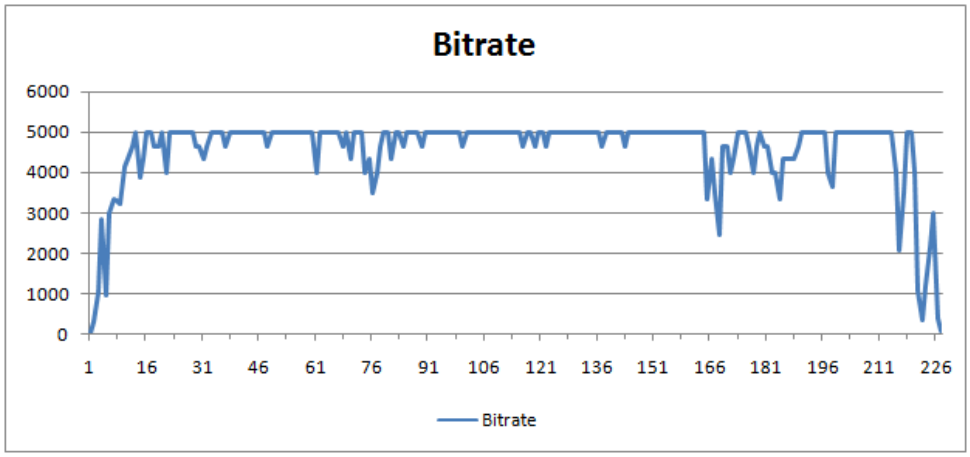


Figura 7.7: Bitrate medio de las pruebas con segmentos de 2 segundos del vídeo Of Forest and Men

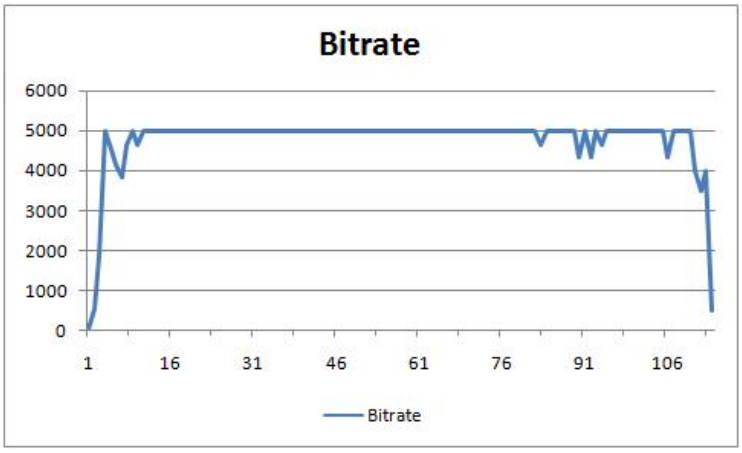


Figura 7.8: Bitrate medio de las pruebas con segmentos de 4 segundos del vídeo Of Forest and Men

7.2.2. Segmentos de 4 segundos

En este caso (figura 7.8), al inicio se producen algunos cambios de calidad, pero después se mantiene la calidad máxima durante bastante tiempo. Al final la calidad vuelve a bajar bastante. En ninguna de las 3 pruebas se detuvo la reproducción.
En las dos primeras pruebas se produjeron alrededor de 15 cambios de calidad, y en la tercera se produjeron 8. Los cambios iniciales y los finales son prácticamente iguales en las tres pruebas, el resto son diferentes.

7.2.3. Segmentos de 6 segundos

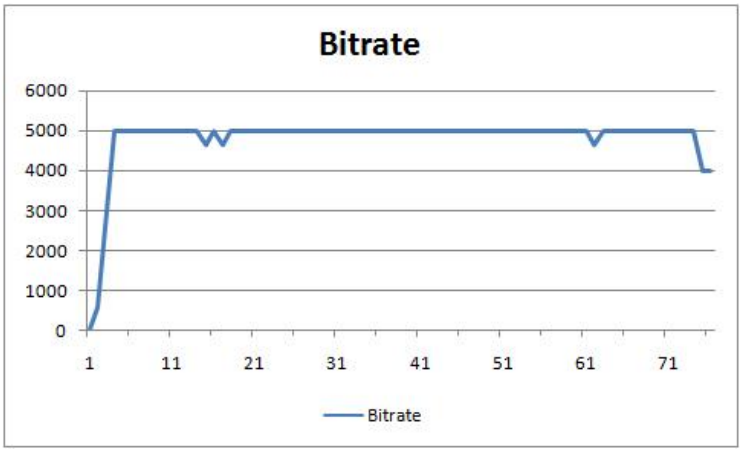


Figura 7.9: Bitrate medio de las pruebas con segmentos de 6 segundos del vídeo Of Forest and Men

En este gráfico (figura 7.9) podemos observar que una vez se alcanza la calidad máxima, se mantiene prácticamente durante toda la reproducción, y en los últimos segmentos baja un poco, pero no tanto como en los casos anteriores. En estas pruebas la reproducción tampoco se ha detenido.
En las tres pruebas se produjeron alrededor de 7 cambios de calidad, siendo exactamente los mismos cambios iniciales y el mismo cambio al final en las tres pruebas. En la primera prueba no se produjeron más cambios y en las otras dos se produjeron algunos cambios intermedios más.

7.2.4. Segmentos de 10 segundos

En este caso (figura 7.10), una vez se alcanza la calidad máxima, se mantiene durante toda la reproducción. En las 3 pruebas la reproducción tampoco se detuvo.
En las tres pruebas únicamente se produjeron cambios de calidad al inicio de la reproducción, hasta que se alcanzó la calidad máxima. Estos cambios fueron exactamente los

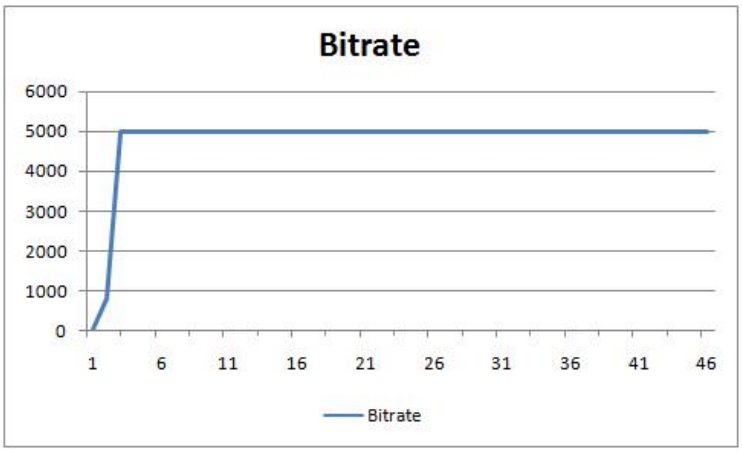


Figura 7.10: Bitrate medio de las pruebas con segmentos de 10 segundos del vídeo Of Forest and Men

mismos en las dos últimas pruebas y en la primera se produjo un cambio diferente en el segundo segmento.

7.2.5. Segmentos de 15 segundos

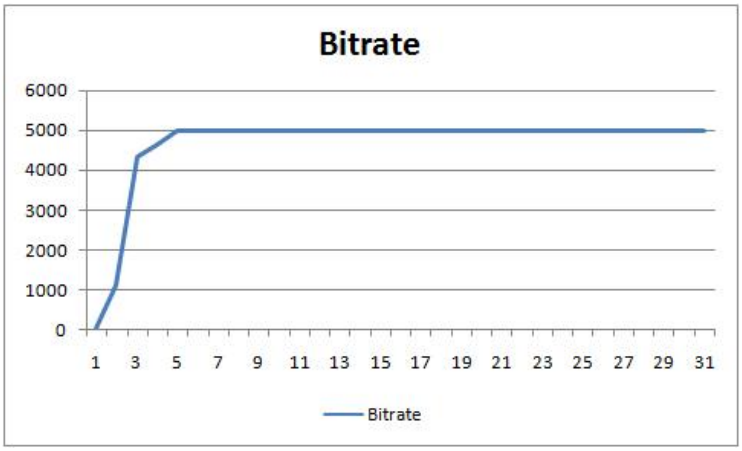


Figura 7.11: Bitrate medio de las pruebas con segmentos de 15 segundos del vídeo Of Forest and Men

Como podemos ver en este gráfico (figura 7.11), los resultados de estas pruebas prácticamente son los mismos que en el caso anterior, y la reproducción tampoco se ha detenido en ninguna de las pruebas.
En las tres pruebas únicamente se produjeron cambios de calidad al inicio de la reproducción, hasta que se alcanzó la calidad máxima, pero en este caso los cambios fueron

diferentes en las tres pruebas.

7.2.6. Conclusiones

Viendo los gráficos de estas pruebas podemos concluir lo mismo que con las pruebas de Big Buck Bunny, que a mayor duración de los segmentos, menos cambios de calidad se producen y se mantiene durante más tiempo la calidad máxima, y que los cambios no siempre se producen en los mismos segmentos aunque en algunas ocasiones sí.

7.3. Valkaama

Finalmente se ha realizado una prueba de la película Valkaama, para realizar alguna prueba con un vídeo un poco más largo. En este caso los bitrates y sus correspondientes resoluciones son los mismos que en el caso de Of Forest and Men.

En el gráfico de la figura 7.12 podemos observar que durante la reproducción se producen algunos cambios de calidad, pero no demasiados en relación a la duración del vídeo (1 hora y media, aproximadamente). La mayoría del tiempo la calidad de los segmentos es la máxima, y la reproducción únicamente se ha detenido un par de veces alrededor del minuto 2, el resto del vídeo se ha reproducido correctamente.

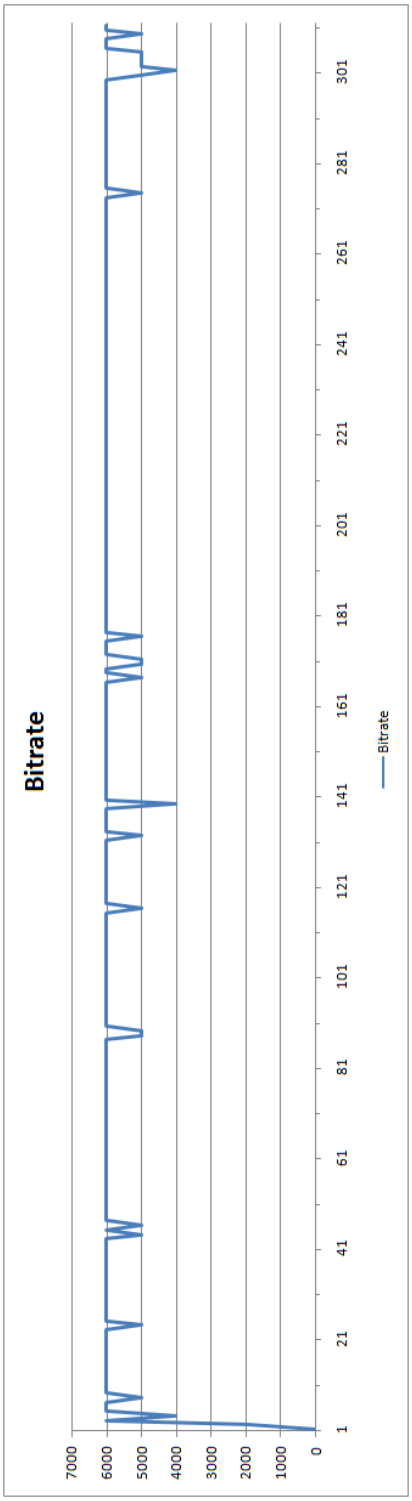


Figura 7.12: Bitrate de una prueba de Valkaama con segmentos de 15 segundos

Parte III

Planificación y conclusiones

Capítulo 8

Planificación

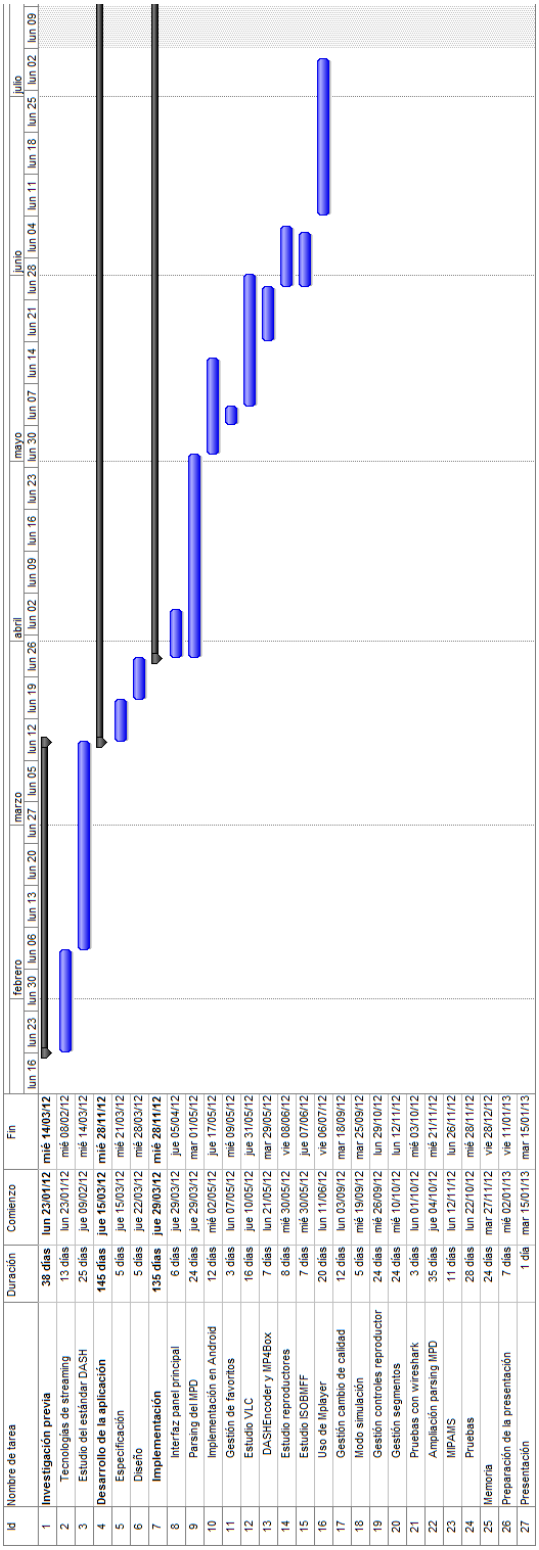
En este capítulo podremos ver las tareas que se han realizado en el proyecto y el tiempo que se ha dedicado a cada una de ellas. Para visualizar mejor la distribución en el tiempo podemos observar el diagrama de Gantt, que está dividido en dos figuras (8.1 y 8.2).

A continuación describiremos brevemente las tareas que aparecen en el diagrama.

- **Investigación previa:** Al inicio del proyecto se realizó una pequeña investigación sobre algunas de las tecnologías de streaming existentes y sobre algunos servidores para ver qué protocolos eran soportados por la mayoría de servidores y también para ver los formatos multimedia soportados.
  Posteriormente se pasó a estudiar en profundidad el estándar sobre DASH para poder desarrollar la aplicación.
- **Desarrollo de la aplicación:** una vez finalizada la fase de investigación se pasó a realizar la especificación y el diseño de la aplicación y posteriormente se inició la implementación.
  A continuación detallaremos algunos aspectos de la fase de implementación:

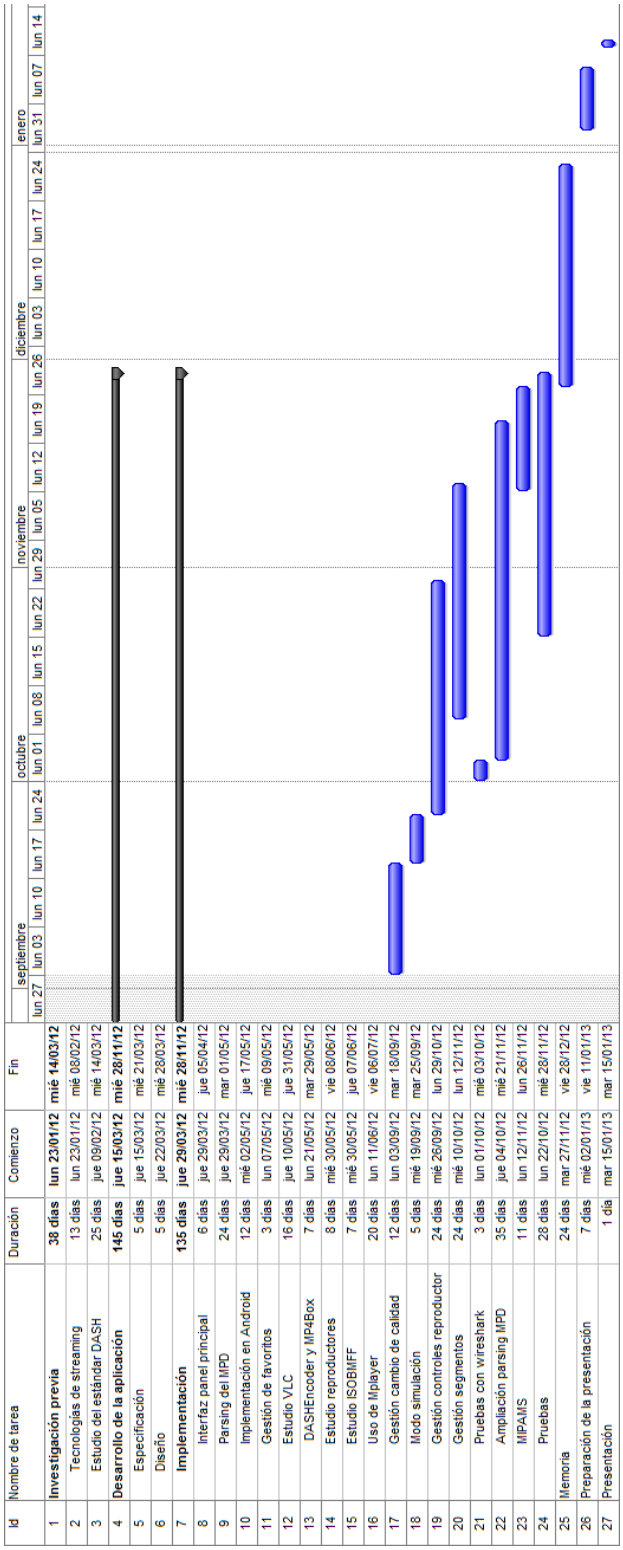
  - Durante las primeras semanas se creó la interfaz del panel principal de la aplicación, al mismo tiempo que se realizaba el parsing del MPD. Esta última tarea tenía el objetivo de obtener los datos necesarios para obtener las urls de los segmentos de forma que se pudieran ir realizando peticiones HTTP para descargar estos segmentos, simulando el comportamiento de una reproducción, aunque todavía no se utilizaba ningún reproductor.

  - A continuación se inició la implementación de la aplicación pero en Android, ya que el objetivo inicial del proyecto era realizar ambas aplicaciones, pero por problemas con los formatos de vídeo en Android finalmente se decidió realizar únicamente la aplicación de escritorio.



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Investigación previa**	**38 días**	**lun 23/01/12**	**mié 14/03/12**
2	Tecnologías de streaming	13 días	lun 23/01/12	mié 08/02/12
3	Estudio del estándar DASH	25 días	jue 09/02/12	mié 14/03/12
4	**Desarrollo de la aplicación**	**145 días**	**jue 15/03/12**	**mié 28/11/12**
5	Especificación	5 días	jue 15/03/12	mié 21/03/12
6	Diseño	5 días	jue 22/03/12	mié 28/03/12
7	**Implementación**	**135 días**	**jue 29/03/12**	**mié 28/11/12**
8	Interfaz panel principal	6 días	jue 29/03/12	jue 05/04/12
9	Parsing del MPD	24 días	jue 29/03/12	mar 01/05/12
10	Implementación en Android	12 días	mié 02/05/12	jue 17/05/12
11	Gestión de favoritos	3 días	lun 07/05/12	mié 09/05/12
12	Estudio VLC	16 días	jue 10/05/12	jue 31/05/12
13	DASHEncoder y MP4Box	7 días	lun 21/05/12	mar 29/05/12
14	Estudio reproductores	8 días	mié 30/05/12	vie 08/06/12
15	Estudio ISOBMFF	7 días	mié 30/05/12	jue 07/06/12
16	Uso de Mplayer	20 días	lun 11/06/12	vie 06/07/12
17	Gestión cambio de calidad	12 días	lun 03/09/12	mar 18/09/12
18	Modo simulación	5 días	mié 19/09/12	mar 25/09/12
19	Gestión controles reproductor	24 días	mié 26/09/12	lun 29/10/12
20	Gestión segmentos	24 días	mié 10/10/12	lun 12/11/12
21	Pruebas con wireshark	3 días	lun 01/10/12	mié 03/10/12
22	Ampliación parsing MPD	35 días	jue 04/10/12	mié 21/11/12
23	MP4MS	11 días	lun 12/11/12	lun 26/11/12
24	Pruebas	28 días	lun 22/10/12	mié 28/11/12
25	Memoria	24 días	mar 27/11/12	vie 28/12/12
26	Preparación de la presentación	7 días	mié 02/01/13	vie 11/01/13
27	Presentación	1 día	mar 15/01/13	mar 15/01/13

Figura 8.1: Diagrama de Gantt (parte 1)



Id	Nombre de tarea	Duración	Comienzo	Fin
1	**Investigación previa**	**38 días**	**lun 23/01/12**	**mié 14/03/12**
2	Tecnologías de streaming	13 días	lun 23/01/12	mié 08/02/12
3	Estudio del estándar DASH	25 días	jue 09/02/12	mié 14/03/12
4	**Desarrollo de la aplicación**	**145 días**	**jue 15/03/12**	**mié 28/11/12**
5	Especificación	5 días	jue 15/03/12	mié 21/03/12
6	Diseño	5 días	jue 22/03/12	mié 28/03/12
7	**Implementación**	**135 días**	**jue 29/03/12**	**mié 26/11/12**
8	Interfaz panel principal	6 días	jue 29/03/12	jue 05/04/12
9	Parsing del MPD	24 días	jue 29/03/12	mar 01/05/12
10	Implementación en Android	12 días	mié 02/05/12	jue 17/05/12
11	Gestión de favoritos	3 días	lun 07/05/12	mié 09/05/12
12	Estudio VLC	16 días	jue 10/05/12	jue 31/05/12
13	DASHEncoder y MP4Box	7 días	lun 21/05/12	mar 29/05/12
14	Estudio reproductores	8 días	mié 30/05/12	vie 08/06/12
15	Estudio ISOBMFF	7 días	mié 30/05/12	jue 07/06/12
16	Uso de Mplayer	20 días	lun 11/06/12	vie 06/07/12
17	Gestión cambio de calidad	12 días	lun 03/09/12	mar 18/09/12
18	Modo simulación	5 días	mié 19/09/12	mar 25/09/12
19	Gestión controles reproductor	24 días	mié 26/09/12	lun 29/10/12
20	Gestión segmentos	24 días	mié 10/10/12	lun 12/11/12
21	Pruebas con wireshark	3 días	lun 01/10/12	mié 03/10/12
22	Ampliación parsing MPD	35 días	jue 04/10/12	mié 21/11/12
23	MPAMS	11 días	lun 12/11/12	lun 26/11/12
24	Pruebas	28 días	lun 22/10/12	mié 28/11/12
25	Memoria	24 días	mar 27/11/12	vie 28/12/12
26	Preparación de la presentación	7 días	mié 02/01/13	vie 11/01/13
27	Presentación	1 día	mar 15/01/13	mar 15/01/13

Figura 8.2: Diagrama de Gantt (parte 2)

- A la vez que se investigaba sobre Android para intentar solucionar los problemas existentes, se añadió a la aplicación de escritorio una lista de favoritos, tal como ya se ha comentado en capítulos anteriores.
- Posteriormente se empezó a estudiar el código de DASH en VLC, tal como ya se comentó en 6.1, a la vez que se investigó sobre las herramientas DASHEncoder (desarrollada por el grupo de Klagenfurt) y MP4Box (de GPAC), que sirven para segmentar vídeos y generar segmentos compatibles para DASH.
- Dado que el estudio de VLC era muy complicado se estudió algo sobre el formato ISOBMFF a ver si ayudaba a conseguir que se pudiese utilizar VLC. Al mismo tiempo se empezó a investigar otros reproductores para encontrar alguno que se pudiera utilizar en la aplicación, y finalmente se encontró MPlayer.
- Después de estudiar el uso de MPlayer se empezó a añadir a la aplicación el código necesario para poder utilizar este reproductor.
- A continuación se implementó el cambio de calidad automático y después el modo simulación, que ya hemos visto en el capítulo 6. También se implementaron los controles del reproductor y la gestión de los cambios de segmento para que el usuario pudiese cambiar de posición durante la reproducción utilizando el slider.
  A continuación se hicieron algunas pruebas con Wireshark para observar el comportamiento de la red al realizar las peticiones HTTP para obtener los segmentos.
- Más tarde, cuando lo que ya se había implementado funcionaba bien, se amplió el parsing del MPD, añadiendo nuevos elementos y atributos que antes no se utilizaban, como por ejemplo el elemento SegmentTemplate, y posteriormente ampliando la gestión de los streams para poder reproducir MPDs que contengan dos streams (uno de audio y uno de vídeo).
- A la vez que se finalizaba la ampliación del parsing del MPD, se implementó la conexión con MIPAMS, para realizar el login y comprobar si el usuario está autorizado para obtener el MPD que ha seleccionado. Para realizar esto antes se estudió el código de esta plataforma y cómo poder registrar los objetos y crear las licencias para poder autorizar a los usuarios.
- Finalmente, durante las últimas semanas de la implementación, se fueron realizando pruebas para ir comprobando el funcionamiento de los cambios que se iban realizando. A pesar de todo, realmente se han realizado pruebas prácticamente desde el inicio de la reproducción, pero se puede decir que las más importantes son las del final ya que es donde se han comprobado más funcionalidades.

- **Memoria:** Durante los últimos días de la realización de pruebas se empezó a redactar la memoria.
- **Presentación:** Finalmente durante las últimas semanas del proyecto, y una vez finalizada la memoria, se realizará la preparación de la presentación.

Capítulo 9

Costes

Después de ver la planificación, en este capítulo analizaremos los costes de la realización del proyecto.

Este proyecto se ha desarrollado bajo una beca del grupo DMAG, cuya duración ha sido de diez meses, realizando cuatro horas diarias, y con una remuneración aproximada de 6,25 €/hora. De esta forma si tenemos en cuenta estos datos, el coste del proyecto ha sido:

80 horas/mes * 10 meses * 6,25 €/hora = 5000 €

Pero si queremos calcular el coste real del proyecto hay que tener en cuenta el coste del personal, del hardware y del software. Para calcular el coste del personal también habría que tener en cuenta que habrían participado diferentes personas con diferentes roles. A continuación podemos ver cual hubiese sido el coste real del proyecto si tenemos en cuenta todos estos factores.

9.1. Coste del personal

Para calcular el coste del personal hemos supuesto que en el proyecto participarían personas con los roles de Analista/Diseñador y de Programador, con un salario aproximado de 30 € y de 20 € respectivamente.
El Analista/Diseñador se encargaría de las fases de Especificación y Diseño, y el Programador, de la fase de implementación y documentación. En la tabla 9.1 podemos ver el coste total.

Rol	**Horas**	**Coste**
Analista/Diseñador	40 horas (10 días, 4 h/día)	1200 €
Programador	636 horas (159 días, 4h/día)	12720 €
Total	676 horas	13920 €

Tabla 9.1: Coste del personal

9.2. Coste del hardware y del software

Para calcular el coste del hardware y del software únicamente deberemos tener en cuenta el coste de un ordenador y de la licencia de Microsoft Windows 7 Professional, ya que el resto de software utilizado es open source y no se ha utilizado ningún hardware más. En la tabla 9.2 podemos ver el coste total.

Recurso	**Coste**
Ordenador gama media	600 €
Microsoft Windows 7 Professional	215 €
Total	815 €

Tabla 9.2: Coste del hardware y del software

9.3. Coste total del proyecto

Para calcular el coste total del proyecto tenemos que sumar los costes que acabamos de calcular en los apartados anteriores. En la tabla

Concepto	**Coste**
Coste del persopal	13920 €
Coste del hardware y del software	815 €
Total	14735 €

Tabla 9.3: Coste total del proyecto

Por tanto si este proyecto se realizase en una empresa, el coste aproximado sería de 14735 €.

Capítulo 10

Conclusiones

Después de haber visto las distintas fases del desarrollo del proyecto, en este capítulo veremos qué objetivos se han cumplido.

El objetivo principal del proyecto, desarrollar una implementación parcial de DASH, se ha cumplido, ya que se ha desarrollado una aplicación de escritorio que permite reproducir el contenido multimedia de un MPD. Además la aplicación cumple con las características que se indicaron en el apartado 1.2, que son:

- Login en MIPAMS y comprobación de permisos para reproducir contenido de un MPD.
- Parsing del MPD.
- Reproducción del contenido multimedia del MPD.
- Cambio de calidad automático durante la reproducción.
- Simulación de cambios en la red.

Tal como ya avanzamos en 1.2, el segundo objetivo, desarrollar una aplicación para DASH en la plataforma Android, no se ha cumplido, debido a problemas con los formatos de vídeo en esta plataforma. En concreto lo que ocurría era que cuando se concatenaba el segmento de inicialización con otro segmento (de la misma forma que se realiza en la aplicación de escritorio) y se le pasaba al reproductor de Android (MediaPlayer), éste no respondía y se volvía a la pantalla anterior. En cambio si se intentaba reproducir un vídeo 3gp no había ningún problema. Por tanto parecía que el problema era el formato de los segmentos.
En la figura 10.1 podemos ver un fragmento del log de una prueba que intenta reproducir un segmento de un MPD concatenado con su segmento de inicialización. Como hemos comentado, no se llega a reproducir nada y se vuelve a la pantalla desde la cual hemos intentado iniciar la reproducción.

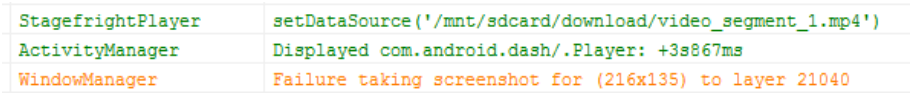


Figura 10.1: Fragmento del log de una prueba con un segmento concatenado con su segmento de inicialización

En la figura 10.2 podemos ver un fragmento del log de una prueba en la que se reproduce un vídeo 3gp. En este caso se reproduce el vídeo con éxito y una vez finaliza, vuelve a la pantalla desde la que se inició la reproducción.

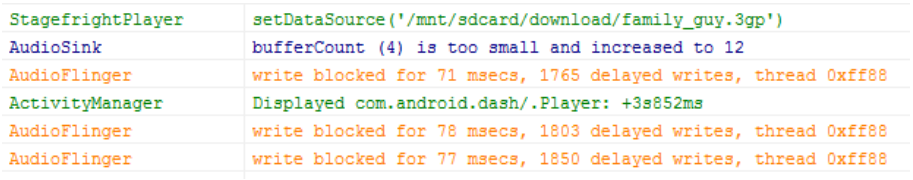


Figura 10.2: Fragmento del log de una prueba con un vídeo 3gp

Después de investigar durante unos días y no conseguir solucionar este problema se decidió centrarse en la aplicación de escritorio y dejar este segundo objetivo como trabajo futuro.

Capítulo 11

Trabajo futuro

En este capítulo explicaremos algunas de las posibles ampliaciones que se podrían realizar.

En primer lugar se podría ampliar la aplicación de escritorio reconociendo más atributos y/o elementos de DASH, ya que en la aplicación que se ha desarrollado no se reconocen todos los elementos y atributos del estándar, sólo se reconoce un subconjunto de ellos.

Otra posible mejora sería utilizar otro reproductor en lugar de MPlayer, ya que éste presenta algunos problemas:

- En Windows, entre segmento y segmento se produce un pequeño parpadeo, cosa que no ocurre en Linux. Esto es debido a que la opción -fixed-vo de MPlayer, parece que no acaba de funcionar bien en Windows.
- Cuando el MPD contiene dos streams multimedia (audio y vídeo), se crean dos procesos de MPlayer, uno para el audio y otro para el vídeo. El de vídeo funciona bien tanto en Windows como en Linux (excepto lo comentado en el anterior punto). En cambio en Linux, entre segmento y segmento, el audio se para durante un pequeño periodo, de forma que se produce un delay entre el audio y el vídeo. En Windows la reproducción de audio parece fluida, sin embargo a medida que se avanza en la reproducción se produce un delay de unos 2-3 segundos entre el audio y el vídeo.
  En definitiva se necesitaría mejorar el delay que se produce entre audio y vídeo, en el caso de que el MPD sea de dos streams.

También se podría mejorar la gestión el cambio de calidad, analizando mejor los cambios en la red de forma que la reproducción sea siempre fluida y no se detenga en ningún momento.
Finalmente, tal como hemos comentado en el capítulo anterior, queda pendiente como trabajo futuro finalizar la aplicación para Android que se inició pero no se pudo finalizar por problemas con los formatos de vídeo.

Índice de figuras

Índice de tablas

Bibliografía

[1] Adaptive bitrate streaming. http://en.wikipedia.org/wiki/Adaptive_bitrate_streaming.

[2] Big Buck Bunny. http://www.bigbuckbunny.org/. 67, 75, 100

[3] Compliance Procedures for Dynamic Adaptive Streaming over HTTP (DASH). http://kth.diva-portal.org/smash/get/diva2:470201/FULLTEXT01. 53

[4] DASH Dataset. http://www-itec.uni-klu.ac.at/dash/?page_id=207. 53, 67, 75

[5] Dynamic Adaptive Streaming over HTTP (DASH). http://www.slideshare.net/christian.timmerer/dynamic-adaptive-streaming-over-http-dash.

[6] Dynamic Adaptive Streaming over HTTP Dataset. http://www-itec.uni-klu.ac.at/bib/files/p89-lederer.pdf. 53, 67

[7] Dynamic Adaptive Streaming over HTTP: From Content Creation to Consumption. http://www.slideshare.net/christian.timmerer/dynamic-adaptive-streaming-over-http-from-content-creation-to-consumption.

[8] El uso de YouTube en cifras. http://blog.posicionamientoconvideomarketing.com/el-uso-de-youtube-en-cifras-una-vez-mas/. 15, 99

[9] Global Internet Phenomena Report. http://www.sandvine.com/downloads/documents/Phenomena_1H_2012/Sandvine_Global_Internet_Phenomena_Report_1H_2012.pdf. 11, 99

[10] GPAC –DASH Sequences. http://gpac.wp.mines-telecom.fr/2012/02/23/dash-sequences/. 67

[11] Historia del Software: Música y vídeo en streaming. http://bitelia.com/2012/02/historia-del-software-musica-y-video-en-streaming.

[12] HTTP Streaming: What You Need to Know. http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/HTTP-Streaming-What-You-Need-to-Know-65749.aspx.

[13] Implementación de RTSP: Real Time Streaming Protocol. http://profesores.elo.utfsm.cl/~agv/elo323/2s10/projects/ApablazaBustamante/index.html.

[14] Information technology –Dynamic adaptive streaming over HTTP (DASH) –Part 1: Media presentation description and segment formats. 21

[15] ITEC –Dynamic Adaptive Streaming over HTTP. http://www-itec.uni-klu.ac.at/dash/. 53

[16] Java y los ficheros .properties. http://www.v3rgu1.com/blog/476/2011/programacion/java-y-los-ficheros-properties/.

[17] Java y XML. http://www.latascadexela.es/2008/07/java-xml.html.

[18] JMPlayer –Embedding MPlayer in Java. http://beradrian.wordpress.com/2008/01/30/jmplayer/.

[19] MPEG-DASH: The Standard for Multimedia Streaming Over Internet.

[20] MPEG's Dynamic Adaptive Streaming over HTTP (DASH) –Enabling Formats for Video Streaming over the Open Internet. http://tech.ebu.ch/docs/events/webinar043-mpeg-dash/presentations/ebu_mpeg-dash_webinar043.pdf. 22, 23, 35, 37, 99

[21] MPlayer. http://www.mplayerhq.hu.

[22] Real Time Messaging Protocol. http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol. 19, 99

[23] Real Time Streaming Protocol. http://es.wikipedia.org/wiki/Real_Time_Streaming_Protocol.

[24] Real-time Transport Protocol. http://en.wikipedia.org/wiki/Real-time_Transport_Protocol.

[25] RTP Control Protocol. http://en.wikipedia.org/wiki/RTP_Control_Protocol.

[26] RTSP (Real Time Streaming Protocol) en IPv4 e IPv6. http://www.cudi.edu.mx/primavera_2007/presentaciones/gestion_fabian.pdf.

[27] Streaming. http://en.wikipedia.org/wiki/Streaming_media.

[28] What Is a Streaming Media Protocol? http://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-Is-a-Streaming-Media-Protocol-84496.aspx.

[29] What Is Streaming? http://www.streamingmedia.com/Articles/Editorial/What-Is-.../What-is-Streaming-74052.aspx.