UNIVERSIDAD CARLOS III DE MADRID
ESCUELA POLITÉCNICA SUPERIOR

GRADO INGENIERÍA TELEMÁTICA

TRABAJO FIN DE GRADO

DISEÑO Y EVALUACIÓN DE UN SISTEMA DE DISTRIBUCIÓN DE CONTENIDO PARA VÍDEO ADAPTATIVO SOBRE ANDROID

Autor: Daniel Martínez Ortiz
Director: Jaime José García Reinoso

<Fecha>

Título:
Diseño y evaluación de un sistema de distribución de contenido para vídeo adaptativo sobre Android

Autor:
Daniel Martínez Ortiz

Director:
Jaime José García Reinoso

Realizado el acto de defensa y lectura del Trabajo Fin de Grado el día ……*04 de Marzo del 2014*…… en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, el tribunal:

Presidente: *Alberto García Martínez*

Secretario: *José Joaquín Escudero Garzas*

Vocal: *Alberto Cortés Martín*

acuerdan otorgarle la calificación de:

Calificación:

<04/03/2014>

A mi hermana, padres y familia,
y toda aquella gente que ha hecho que
pueda llegar hasta aquí con su apoyo.

Agradecimientos

A todos esos compañeros de clase tanto de la universidad como del colegio, gracias a los cuales he mejorado y crecido como persona. Haciendo principal referencia con los que más he colaborado durante mi crecimiento académico. José, con el que he pasado mayores dificultades ante los baches que surgen en la carrera; Víctor, gracias al cual he consolidado una gran amistad; Rodrigo, que me enseñó como apreciar a cada persona tanto por sus virtudes como defectos; y Dani, que me aporto un cariño especial por los libros. Y a mis compañeros de Orange que me aportaron una visión profesional diferente a la que otras personas me han podido hacer ver durante mi crecimiento académico y profesional. Y a mis chicos de Cieza, con los que he compartido viajes y experiencias inolvidables.

Agradecer también a mi tutor Jaime todo el tiempo dedicado para poder realizar este proyecto y su gran dedicación para ayudarme.

Sin tampoco olvidarme de las dos personas más importantes que me han hecho crecer como persona tanto en el ámbito personal como académico, mejorando y apoyándome día a día. Mis dos mejores amigos Carlos y Alberto, aunque me cueste referenciarme al segundo de este modo. Gracias por todos esos momentos de máxima intensidad.

Y por último, y no menos importante, sino todo lo contrario. Agradecer a mi increíble familia todo lo que me ha aportado, pues siempre han estado en lo bueno y en lo malo, apoyándome en mis peores momentos y aconsejándome en todo lo posible.

Gracias a todos estos y aquellos que aun no nombrándoos habéis formado parte de mi vida, pues cada uno me ha aportado un pequeño granito de arena para hacerme crecer y ser como soy.

Resumen

Durante los últimos años, la tecnología ha dado pasos agigantados en su evolución. La sociedad vive a un ritmo frenético y exige conexiones más veloces y tiempos de espera cada día menores. Debido a esto, se ha aumentado la velocidad y ancho de banda tanto en redes Wi-Fi como en redes de datos, véase las nuevas incorporaciones al mercado de tecnologías punteras como el 4G y la fibra óptica, que ayudan a los usuarios a adquirir una mayor velocidad en sus conexiones a Internet.

Vivimos en un mundo sin tiempo y con prisas, y por ello, las tecnologías focalizan sus objetivos en cubrir, principalmente, esta demanda de velocidad. La sociedad hace un gran uso de vídeos y fotos en todo momento, toda red social actualmente ya admite tanto imágenes como vídeos para compartir, por ello en este proyecto nos centramos en la importancia de los vídeos.

La existencia de un aumento en el tráfico de streaming en Internet hace que los usuarios requieran mayor velocidad de carga en vídeos. Además, el número de dispositivos desde los cuales podemos reproducir estos contenidos ha incrementado gracias a smartphones, tablets, smart TV's, etc., de manera que podemos ver desde cualquier lugar nuestros vídeos favoritos a través de conexiones 3G, 4G o incluso Wi-Fi. Sin embargo, no todos los dispositivos pueden reproducir todo lo que desearíamos, debido a la existencia de diferentes protocolos de streaming, diferentes formatos, etc., que hacen que no todos los dispositivos sean compatibles con estas especificaciones. Para resolver estos problemas, hemos utilizado en este proyecto un nuevo estándar que permite reproducir contenido multimedia en streaming adaptado a cualquier dispositivo y red de comunicaciones, conocido como DASH (Dynamic Adaptive Streaming over HTTP). Una vez solucionado el primer problema, se quiso indagar en cómo hacer que estos vídeos streaming tuviesen un menor retardo y tiempo de carga.

Basándose en estas necesidades, se ha realizado este Trabajo Fin de Grado, en el que se ha implementado y diseñado un sistema para el análisis y funcionamiento de las redes centradas en el contenido (CCN) en un terminal Android. El sistema consiste en crear una red virtual basada en CCNx que nos permite estudiar mediante el estándar DASH, cómo se comporta un terminal Android en la reproducción de vídeos por streaming en redes Wi-Fi. Gracias a las redes CCN conseguiremos un tiempo de carga menor ya que en estas redes se almacenarán los fragmentos de vídeo que pasen por las mismas y ya no siempre será necesario conectar con el servidor.

De esta manera, este proyecto se puede dividir en diferentes etapas. La primera de ellas consiste en, por un lado, la creación de ficheros de metadatos que describen el contenido disponible, entre otras características, (ficheros Media Presentation Description, MPD) para la reproducción de los vídeos. Por otro lado, la búsqueda de herramientas que manejen estos ficheros. Para ello se pensó en el reproductor VLC, pero finalmente se decantó por el reproductor multimedia para Android, Osmo4, debido a que es de código abierto y permite una mayor facilidad en su configuración. La segunda parte, es la configuración de la red CCNx que nos permitirá almacenar las

trazas del vídeo en una caché para que el cliente adquiera el vídeo en menor tiempo sin tener que pasar por el servidor. Por último, se realizó una batería de pruebas mediante un emulador y un terminal Android (HTC Legend) para finalizar con el estudio. Gracias a estas pruebas podemos demostrar que este tipo de redes centradas en la información, junto con DASH, nos ayudan a recortar tiempos de carga del vídeo, lo que mejora la experiencia final de los usuarios.

Abstract

During years, technology has grown rapidly. Society lives with no time for anything and they demand faster connections and lower waiting times. As a result, as far as bandwidth and Wi-Fi networks is concerned, speed has increased. For example, the new additions of 4G and fiber optics technologies. These technologies help users to have better internet connections.

Technologies are constantly improving to solve these problems. Society makes massive use of videos and photos at any time, even in the social networks that currently support images and videos to share. Therefore, we stress the importance of the videos today.

The existence of an increasing traffic of streaming on the Internet makes users to require fastest access in the videos. In addition, the number of devices that we can reproduce has increased thanks to smartphones, tablets, smart TVs, etc. We can see anywhere our favorite videos with 3G, 4G connections or even Wi-Fi. However, not all devices can reproduce everything because there are different protocols of different formats, streaming, etc. To resolve these problems, a possible solution might be to create a new standard allowing users to play multimedia content streaming from any device. For this reason the standard called DASH (Dynamic Adaptive Streaming over HTTP) was created standard. Once the first problem is solved, we are looking for making these streaming videos to have faster loading time. For this reason, we use CCNx networks that allow us a lower playback time, because of its capability to cache video frames at the own routers.

Based on these needs has been carried out this thesis, in which we have designed a system for analysis and operation of networks CCN on an Android device. The system consists of a virtual network based on CCNx that allows us to study how it plays an Android device videos streaming according to the DASH standard. With CCNx networks we will get less load time, because the video frames that pass through it will be stored there, reducing the number of hops to retrieve the content.

This thesis is divided into different parts. The first part consists of the creation of files for metadata that describes the content available, among other characteristics, (Media Presentation Description, MPD files) for the playback of videos. On the other hand, the search for tools that handle these files. We started using VLC Player, but finally we have used the Media Player for Android, Osmo4, since it is open source too and it includes extra functionalities. The second part is the network configuration CCNx allowing us to store traces of the video in a buffer so that the client has the video in less time. Finally, a battery of tests using a simulator of Android and HTC Legend terminal was to end with the study. These tests can show if such networks help us to have less load times or, however, its improvements are not good.

Índice general

Índice de figuras y gráficas

Parte 1
Introducción

Este documento describe el proyecto realizado como Trabajo Fin de Grado, el cual consiste en el diseño y estudio del comportamiento de un sistema de distribución para vídeo streaming en Android sobre redes CCN (Content-Centric Networks, redes centradas en el contenido) en un entorno Wi-Fi y su correspondiente implementación y despliegue práctico.

A continuación se introducen las tecnologías utilizadas y se destacan los aspectos que han motivado a la realización de dicho estudio.

I Motivación

La principal motivación de este Trabajo Fin de Grado viene dado al gran interés que despiertan últimamente la transmisión de vídeos por streaming. El gran aumento en el consumo de este tipo de vídeos, del que posteriormente se hablará, nos ha llevado a estudiar el comportamiento de los mismos y cómo poder mejorar su capacidad de reproducción en menores tiempos de carga. Como bien se conoce, este

tipo de vídeos se caracteriza debido a que funcionan gracias a que en el lado del cliente se dispone de un buffer de datos que almacena el contenido descargado para, posteriormente, mostrarlo. Debido a la gran demanda existente, se ha investigado en cómo reducir los tiempos de envío de estos segmentos de vídeo para que el usuario disponga de ellos lo antes posible. Muchos de los usuarios que requieren la reproducción de vídeos se ven en la tesitura de necesitar mayores tiempo de espera para contenidos multimedia en una alta calidad, ya que si un usuario recibe un fragmento y el siguiente no (debido a que se requiere mayor tiempo de carga por su elevado peso), el usuario tendrá el vídeo congelado (pausado) hasta que disponga del siguiente fragmento a reproducir. Gracias a nuestra investigación, hemos analizado y estudiado cómo se comportan las redes CCN (las cuales nos permiten almacenar los segmentos solicitados en los nodos de dichas redes) con vídeos streaming (basados en el estándar DASH) que intentan disminuir estos tiempos de carga. En los siguientes puntos se mostrará el resultado de este estudio y si el uso de estas redes mejora estos tiempos para calidades altas de vídeos, teniendo en cuenta el ancho de banda, y adaptando estas pruebas a un dispositivo Android capaz de reproducir el tipo de vídeo demandado siguiendo las pautas requeridas por el estándar DASH.

II Redes inalámbricas

La evolución de los dispositivos móviles que hoy en día abundan en la sociedad se alimentan, básicamente, de las redes inalámbricas, gracias a las cuales los usuarios pueden consultar todo tipo de contenidos online desde cualquier punto. Esto supone una mayor libertad para el usuario a la hora de acceder a todas las funcionalidades que la red ofrece, ya que no hace falta tener que usar únicamente una red WiFi [2].

Según el Informe Cisco VNI [1], en 2017 casi la mitad del tráfico móvil se descargará a las redes Wi-Fi, mientras las redes 4G, actualmente en auge, soportarán cerca del 10 por ciento del total de conexiones móviles. Esto indica cómo poco a poco las redes inalámbricas van alcanzando tasas muy altas de uso. En los próximos cinco años el tráfico global de datos móviles se multiplicará por 13, alcanzando los 11,2 Exabytes mensuales o 134 Exabytes anuales, lo que supone un ratio de incremento interanual del 66 por ciento en dicho período (2012-2017) como se muestra en la *Figura 1*. Un incremento que denota cómo los usuarios cada vez son más proactivos al uso de tecnologías móviles. También se desvela en este informe que solamente el tráfico de datos que se añadirá a Internet móvil en el año 2016 alcanzará los 3,7 Exabytes mensuales, multiplicando por cuatro el tamaño estimado de todo Internet móvil en 2012 (884 Petabytes mensuales) [3]. En la siguiente figura se muestra la tendencia creciente del tráfico móvil durante los próximos cinco años:

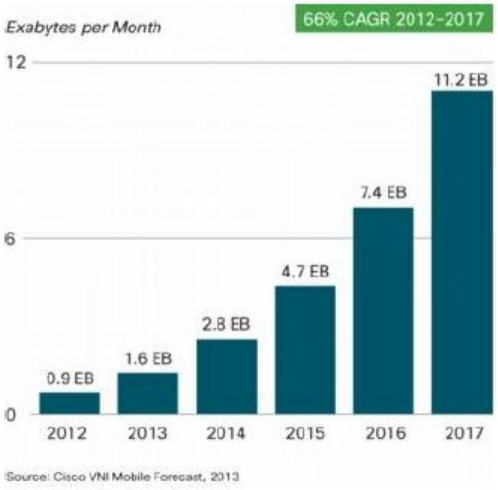


Figura 1 – Previsión tráfico.

En cualquier parte del mundo, la sociedad está consumiendo cada vez más ancho de banda inalámbrica para gestionar una variedad más amplia de tareas. Los datos muestran que los dispositivos móviles se están convirtiendo en una herramienta imprescindible: cada vez hay más gente que paga por aplicaciones, y está más dispuesta a intercambiar su privacidad por beneficios. En la siguiente figura [*Figura 2*] se muestra cómo Internet y los dispositivos móviles crecen juntos.

Figura 2 – Previsión del uso mundial de datos móviles, por región.

Otra de las grandes demandas es el vídeo. Que se ha incrementado notablemente en los últimos años. En el siguiente gráfico se muestra el uso que se hace del tráfico de datos en los tablets, los smartphones y los Mobile PCs [*Figura 3*]. Cabe destacar cómo la gran mayoría de este flujo de datos proviene de los vídeos streaming y la navegación web. Lo que nos lleva a darnos cuenta de la importancia de los mismos en esta sociedad.

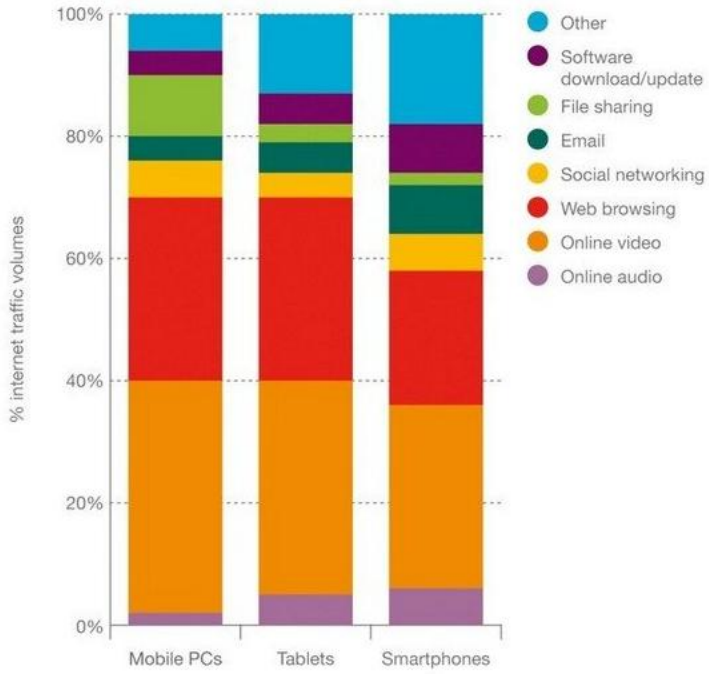


Figura 3 – Uso de tráfico por tipo de dispositivo.

Siguiendo en nuestro interés por los vídeos nos encontramos con que Cisco ha realizado un estudio sobre las previsiones de tráfico móvil de 2012 a 2017 y señala que el vídeo en los dispositivos móviles supondrá dos tercios del tráfico en 2015 [*Figura 4*]. Actualmente representa un 48,5% mientras que en 2017 será de un 66,5%, por lo que se puede comprobar que año tras año incremente su porcentaje [1]. Este crecimiento se puede atribuir al creciente número de usuarios, a las velocidades de conexión cada vez más rápidas y a la mayor autonomía de las baterías. A continuación, se muestran las previsiones para el período 2012-2017 del porcentaje correspondiente al vídeo en los dispositivos móviles:

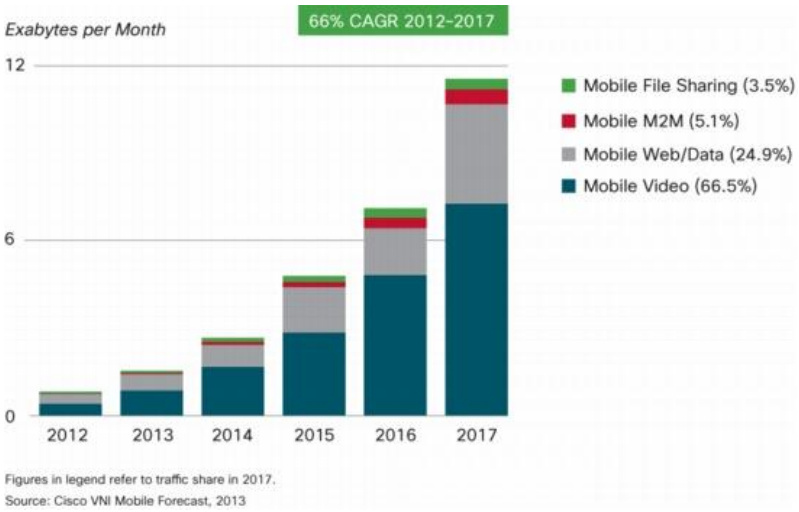


Figura 4 – Uso de tráfico por año.

Podemos concluir así, que el uso del vídeo streaming está teniendo un fuerte impacto en las redes inalámbricas y en el uso de los nuevos usuarios con acceso a la red.

III Streaming

Gran parte de los vídeos que se encuentran en Internet se reproducen en streaming. Este proceso consiste en que un usuario consuma un archivo multimedia a través de una red de datos al mismo tiempo que se descarga. Para ello, esta tecnología requiere de un buffer de datos en el lado del receptor, que almacene todas los segmentos que se vayan descargando para su posterior reproducción.

En los años 90, gran parte de los ordenadores eran capaces de reproducir archivos multimedia. Sin embargo, la descarga de este tipo de ficheros tenía una duración demasiado larga, por lo que resultaba casi imposible la reproducción de los mismos. A pesar de estos problemas, en esa década comenzaron a realizarse eventos en directo a través de la red, esto promovió el lanzamiento del primer reproductor de vídeo en streaming, RealPlayer. Poco a poco fueron saliendo al mercado otros como QuickTime, de Apple, hasta llegar en 2005 al lanzamiento de YouTube, un sitio web donde los usuarios podían subir y visualizar vídeos. En la *Figura 5* se puede visualizar una infografía de algunos datos de YouTube [25]. Más adelante, apareció en 2008 Spotify, una aplicación que reproduce música vía streaming.

Vista la evolución que ha desarrollado, nos vamos a centrar en su funcionamiento:

- **Conexión con el servidor.** El cliente pide al servidor remoto la reproducción del fichero, establece la conexión y el servidor comienza a enviarle dicho archivo.
- **Almacenamiento en buffer.** Una vez el cliente empieza a recibir las diferentes tramas, las almacena en su buffer para ir reproduciéndolas en orden.
- **Reproducción.** Cuando llega la primera trama, comienza la reproducción. Mientras el cliente está reproduciendo el vídeo se siguen realizando las posteriores descargas en segundo plano hasta recibir todos los fragmentos del vídeo.

Si se produjesen descensos en la velocidad de descarga durante la reproducción, el cliente seguiría mostrando el vídeo consumiendo toda la información almacenada en el buffer. En el caso de que se consumiese todo el buffer se detendría el vídeo hasta volver a tener fragmentos que reproducir.

A continuación vamos a hablar de varios de los protocolos más importantes de streaming.

Figura 5 – Infografía sobre YouTube.

III.1 Real-time Transport Protocol (RTP)

Protocolo del nivel de aplicación para la transmisión de información en tiempo real [29]. No se garantiza la llegada de todos los paquetes a su destino debido a que funciona sobre UDP; y se utiliza junto al protocolo RTCP [III.2] y con menor frecuencia junto al protocolo RTSP [III.3]. Funciona tanto en multicast como en unicast y su principal uso es para el transporte de audio y vídeo. En su información más relevante podemos destacar su uso de *timestamps* para intentar la sincronización; para la perdida de paquetes y para reordenar las tramas se utilizan *números de secuencias*, y un *payload* para indicar el formato de codificación [27]. Para cada archivo multimedia establece una sesión, que consiste en una dirección IP con dos puertos para RTP y RTCP [19].

III.2 RTP Control Protocol (RTCP)

Protocolo que proporciona para un flujo RTP información de control. Su principal función es informar de la calidad de servicio (QoS) proporcionada por RTP, aunque se destaca porque no transporta ningún dato por sí mismo [37]. RTCP define varios tipos de paquetes:

- **Sender Report (SR):** Se envía periódicamente por parte del emisor. Su objetivo es informar de las estadísticas de transmisión y recepción de los paquetes RTP enviados.
- **Receiver Report (RR):** Lo utilizan los participantes pasivos para enviar estadísticas sobre la recepción.
- **Application-specific message (APP):** Mecanismo para diseñar extensiones específicas de la aplicación para este protocolo.
- **Source Description (SDES):** Envía el CNAME (especifica que el nombre del dominio es un alias de otro) a los participantes. Otro uso corriente es para enviar otra tipo de información, ya sea el nombre, correo electrónico, número de teléfono o dirección del emisor.
- **End of participation(BYE):** Se utiliza para indicar el fin de la participación en una sesión.

III.3 Real Time Streaming Protocol (RTSP)

Protocolo de nivel de aplicación. Su principal beneficio es que nos permite mantener una o múltiples sesiones para controlar la entrega de datos en tiempo real, tanto audio como vídeo [28]. Su sintaxis y operación son similares a HTTP, con varias diferencias destacables: los datos son transportados en un protocolo diferente, tanto servidor como cliente pueden enviar peticiones, el servidor necesita mantener el estado de la conexión y utiliza *rtsp://* en vez de *http://* [26]. Sus principales propiedades son:

- **Seguridad.** Utiliza los de HTTP y reutiliza mecanismos de seguridad web.
- **Es extensible.** Tiene la capacidad de añadir nuevos parámetros.
- **Capacidad multiservidor.** Los streams pueden localizarse en diferentes servidores.

- **Puede utilizar UDP o TCP.**

En la siguiente figura [*Figura 6*], se muestra los diferentes estados por los que pasa el cliente:

Figura 6 – Diagrama de estados del cliente en sesión RTSP.

* SETUP: Nos informa de cómo debe ser el flujo de datos transportado. La petición contiene un especificador de transporte y la URL del stream multimedia. La respuesta del servidor confirma estos parámetros escogidos.
* PLAY: Nos indica que está todo listo para que comience a enviar los datos el servidor.
* PAUSE: Detiene temporalmente la reproducción.
* TEARDOWN: Finaliza la sesión. Se detiene los streams y se liberan los recursos.

III.4 HTTP Streaming

Tras comentar algunos de los protocolos más utilizados hay que tener presente ciertos inconvenientes. El principal inconveniente es que en Internet y en varios firewalls, muchas redes de distribución de contenidos no admiten RTP. Otro problema es que en RTP el servidor gestiona por cada cliente una sesión diferente, por lo que el servidor siempre recibe una carga elevada. Como hemos visto anteriormente, RTP funciona junto con RTCP y RTSP, por lo que se incrementa drásticamente la información que se necesita transmitir a través de la red.

Una solución a todos estos problemas es utilizar HTTP Streaming, puesto que la gran mayoría de firewalls sí que admiten este tipo de tráfico y también debido a que el cliente no necesita mantener una sesión con el servidor. El problema reside en que existen diferentes implementaciones en las que cada una utiliza diferentes formatos de segmentos y sus ficheros manifest, por lo que si un dispositivo quisiese reproducir contenido multimedia de un servidor deberá soportar el correspondiente protocolo.

Por lo general, una vez los datos han sido enviados al cliente, el servidor web no termina la respuesta. Se deja el canal abierto por el servidor web para que, si un evento ocurre, pueda ser enviado a uno o muchos clientes inmediatamente. Sino ocurriese esto, los datos estarían en una cola hasta la próxima petición del cliente.

IV Objetivos

En esta sección se describirán los objetivos que han llevado a realizar el Trabajo Fin de Grado. En primer lugar se hablará sobre los objetivos que se plantearon al comienzo del mismo y posteriormente sobre los objetivos técnicos a los que poco a poco se quisieron llegar.

IV.1 Objetivos iniciales

El objetivo fundamental de este Trabajo Fin de Grado es el diseño de una red de distribución de contenido que soporte la transmisión de video streaming adaptativo, basado en DASH, sobre redes Wi-Fi. Este diseño debe ser validado mediante una red de pruebas, en donde se realizarán diferentes experimentos con el fin de analizar las prestaciones de la misma en un entorno inalámbrico.

Gracias a estas redes se obtendrá una reproducción de streaming multimedia en un menor tiempo, puesto que una vez que el servidor obtenga la primera petición de reproducción, dichas redes almacenarán los fragmentos que pasen por ella, así la siguiente vez que se quiera reproducir el vídeo no hará falta que el servidor nos lo sirva, sino que la red CCN ya tendrá estos fragmentos y nos los devolverá sin realizar más saltos por la red. Como posteriormente se verá, el descriptor del vídeo (MPD, más adelante profundizaremos en este término) tendrá un formato concreto que contendrá toda la información necesaria para contactar con el servidor, elegir la calidad adecuada y otros detalles que ya se explicarán.

Una vez estudiados y comprendidos estos primeros objetivos, se indagó más en cómo poder desplegar esta red y que, utilizando un terminal Android, se pudiesen reproducir cualquier tipo de vídeo con formato basado en el estándar DASH.

IV.2 Objetivos técnicos

En este apartado se concretan los objetivos más técnicos y puntuales que se fueron planteando a lo largo del proyecto. Se dividen en varios puntos:

- **Conocer las redes CCN.** El primer objetivo que se marcó para el correcto desarrollo del proyecto fue conocer más a fondo el funcionamiento de este tipo de redes. Se indagó a través de la página oficial y otras fuentes de apoyo para comprender cómo funcionaban, comprender su protocolo (protocolo CCNx) y si soportaban el estándar de vídeo DASH. Una vez asegurados de este último se pasó al siguiente punto.
- **Dominio del entorno.** Como ya se ha comentado anteriormente, el estudio se realizará con un dispositivo Android por lo que se deberá de conocer y familiarizar con este lenguaje de programación. Una parte importante de este lenguaje es el SDK (Software Development Kit), que permite probar, construir y depurar aplicaciones, ya que es un conjunto de herramientas de desarrollo de software. Adentrándonos en la arquitectura del sistema operativo y su

distribución de carpetas y archivos, se deberá de tener cierta cautela, puesto que los archivos de la parte lógica, en Java, referencian los de la parte gráfica, en XML, que referencian a iconos o cadenas de caracteres.

- **Emulador.** Una vez se tiene la aplicación corriendo se ha apoyado en el emulador de Eclipse (Android Emulator) y el uso del DDMS (Dalvik Debug Monitor Service) [20] [21]. El DDMS es una utilidad de depuración integrada en Eclipse, consta de cinco partes funcionales [*Figura 7*]: manejo de tareas (1), donde encontraremos los emuladores y teléfonos que tengamos conectados y sus instancias; manejo de archivos (2), interacción con el emulador (3), sistemas de Log (4), y capturas de pantalla (5).

Figura 7 – Entorno Eclipse DDMS y emulador Android.

Gracias a este entorno se podrá navegar a través del sistema de archivos del terminal emulado para conseguir ciertos datos de los que más tarde hablaremos (logs).

- **Análisis y ejecución de pruebas.** Una vez todo dispuesto, el siguiente objetivo fue el estudio y comportamiento de la aplicación sobre la red CCN y si su comportamiento era el esperado. Se analizó minuciosamente a través de diferentes baterías de prueba, de las que más tarde hablaremos, para comprobar si existía una reducción en el tiempo de carga de los streaming sobre estas redes y el comportamiento del mismo sin dicha red.

Parte 2
Estado del arte

En esta parte del documento se hablará de la base teórica sobre la que nos sustentamos. Se hará, por tanto, un repaso de las técnicas relacionadas para conocer los aportes que nos proporciona este proyecto.

En los siguientes apartados se analiza las tecnologías en las que se apoya este proyecto. En primer lugar es necesario presentar y analizar el formato de vídeo que vamos a trabajar y el estándar en el que se apoya, DASH. Posteriormente hablaremos de la aplicación utilizada para la reproducción del mismo y los requisitos necesarios para su correcto funcionamiento en nuestro terminal Android. Por último, expondremos detalladamente en qué consisten las redes CCN.

I Dynamic Adaptive Streaming over HTTP

Como su propio nombre indica, DASH es un estándar para streaming adaptativo sobre HTTP. Estándar desarrollado por MPEG (Moving Picture Expert Group), responsable del desarrollo de gran parte de los estándares multimedia [4].

Observando las perspectivas del mercado de Internet Streaming y por petición de la industria, MPEG desarrolló un estándar para la transmisión multimedia a través de Internet, estándar conocido como MPEG-DASH. MPEG-DASH es un nuevo estándar para la transmisión de contenido multimedia a través de Internet [6].

Para mostrar la importancia que este estándar está teniendo actualmente, se ha recogido información de una de las ferias más importantes para la industria mundial de los medios electrónicos, exhibiendo más de 1.300 expositores, el IBC [36]. La Conferencia IBC es el principal foro para la presentación y discusión de las nuevas tecnologías, los flujos de trabajo, y modelos de negocio. En el IBC 2013 se ha ofrecido muchas demostraciones de este estándar para darlo a conocer al mundo. En el siguiente mapa [*Figura 8*] se muestra las empresas que han ofrecido estas demostraciones y su ubicación dentro del RAI.

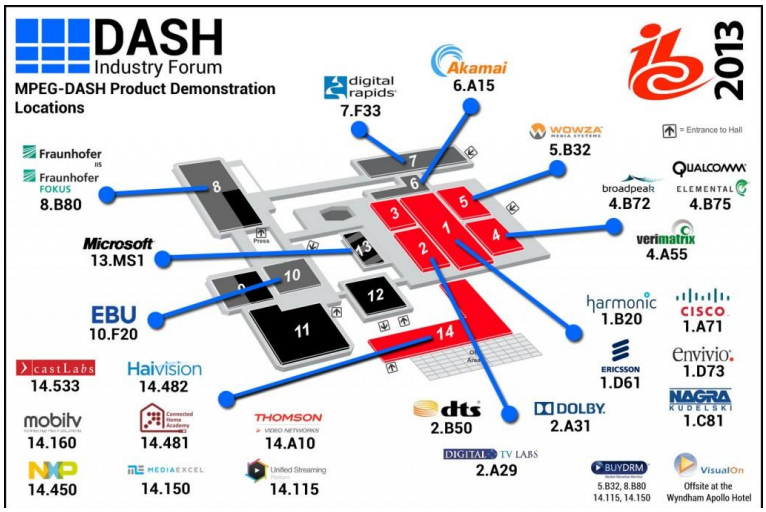


Figura 8 – Plano conferencia IBC 2013.

En este modelo de streaming, todo el control recae únicamente en el cliente. Dicho cliente solicita a servidores web estándar la información necesaria a través del protocolo HTTP. Se deduce de esta manera que, dicho estándar, focaliza su atención en el formato de los datos utilizados.

Vamos a resumir rápidamente el contenido DASH [8]:

- **MPD (Media Presentation Description).** Documento XML, fichero de metadatos, que describe dónde se encuentran, sus diversas alternativas, las direcciones URL, el contenido disponible y otras características.
- **Segmento.** Parte continua de un recurso multimedia. El segmento es la parte más pequeña del vídeo y puede ser localizada en el MPD. Su contenido

depende del formato multimedia subyacente del contenido.

- **Subsegmento.** Parte continua de un segmento o de un subsegmento.
- **SIDX (SegmentIndexBox).** El objetivo de la SIDX es construir un índice del segmento con una granularidad determinada para simplificar los trick modes (búsqueda, avance rápido, retroceso,…).

Para comprender mejor el funcionamiento de este estándar vamos a proceder con un ejemplo muy simple. En la *Figura 9* se ilustra un escenario de transmisión simple entre un servidor HTTP y un cliente DASH. En este ejemplo el contenido multimedia es almacenado y capturado en el servidor HTTP y se entrega también mediante este protocolo (HTTP) [7]. El contenido nos lo podemos encontrar en el servidor en dos partes:

* Media Presentation Description (MPD). Anteriormente definido.
* Segmentos. Contienen las cadenas de bits multimedia reales particionados en uno o varios archivos.

Para la reproducción del contenido, el primer objetivo del cliente DASH es obtener el MPD. Este MPD se puede entregar a través de HTTP (el más común), unidad externa USB, correo electrónico, etc. En el momento en el que el cliente analiza el MPD, aprende acerca de la disponibilidad de medios, tipos de medios, las resoluciones, anchos de banda mínimo y máximo, tiempos, funciones de accesibilidad, etc. Con toda esta información, el cliente DASH escoge la mejor alternativa de codificación, y posteriormente, con esta información, el cliente realiza la búsqueda de estos segmentos utilizando solicitudes HTTP GET.

Posteriormente, se proveerá de un buffer que permita adaptarse a las variaciones de rendimiento de red y así poder controlar las fluctuaciones de ancho de banda de la red. Dependiendo de sus cálculos, el cliente decidirá cómo adaptarse al ancho de banda disponible.

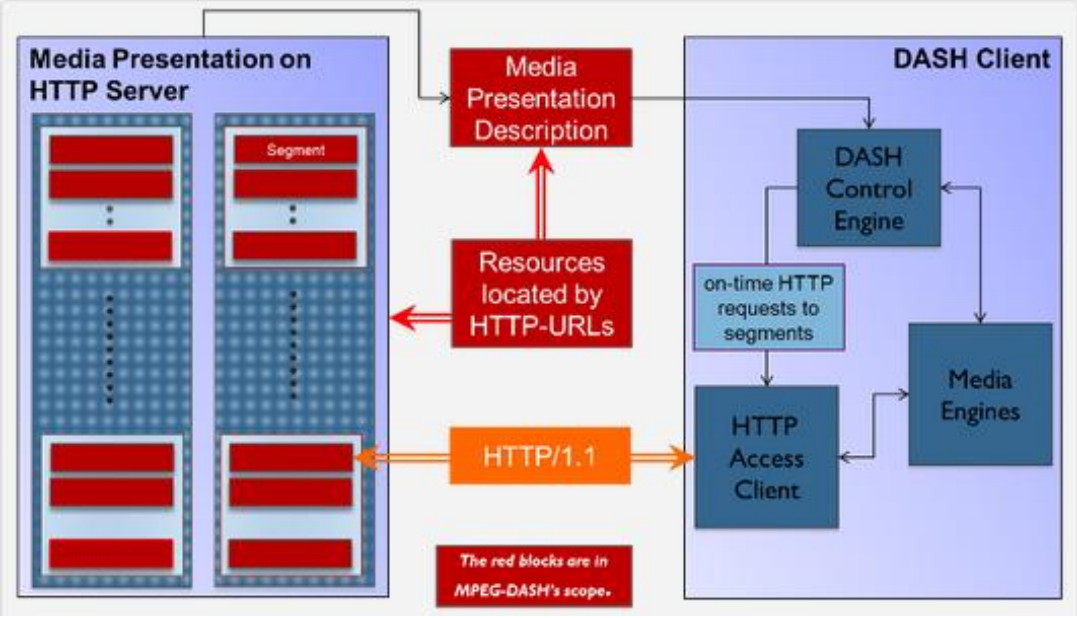


Figura 9 – Escenario de transmisión con cliente DASH.

I.1 Media Presentation Description (MPD)

Media Presentation Description (MPD) es un fichero XML que está en consonancia con el esquema XML definido por las especificaciones 3GPP Adaptive HTTP Streaming [9]. Este documento XML contiene los metadatos necesarios para que se pueda acceder a los segmentos y proporcionar así un servicio de streaming al usuario.

A continuación se muestra el modelo de datos de un fichero MPD [*Figura 10*], posteriormente se irá definiendo cada modelo jerárquico uno a uno.

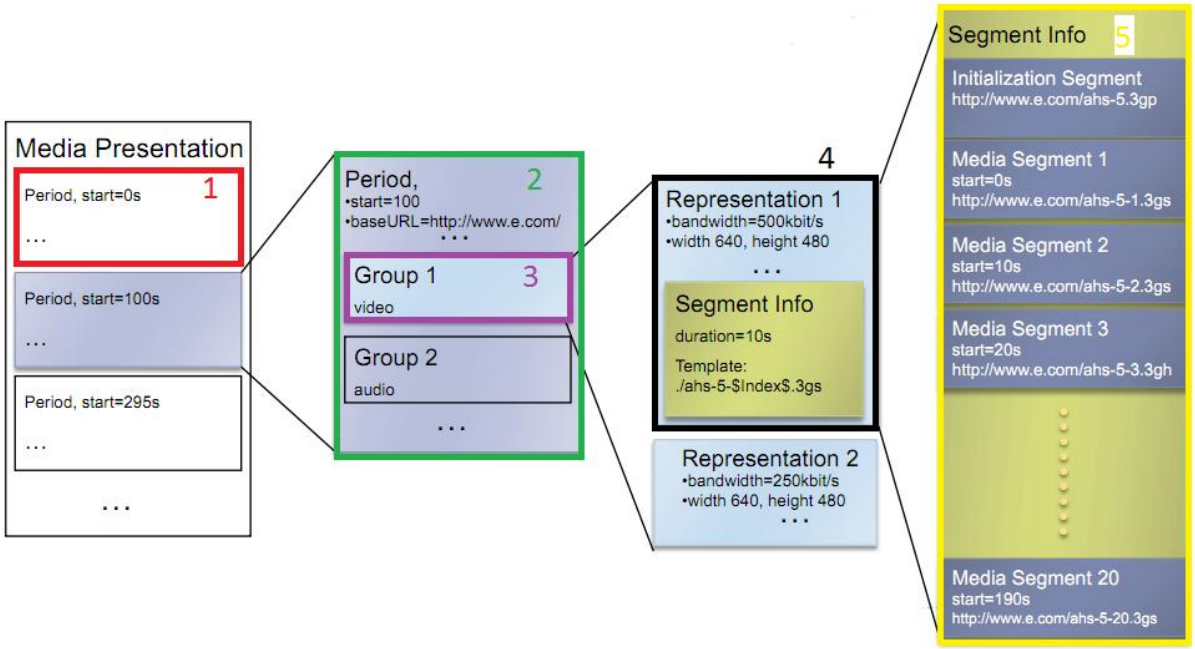


Figura 10 – Modelo de datos MPD.

- **(1)y(2) Period:**

Cada MPD consta de uno o varios Periods. Un Period es un intervalo en el cual se encuentra nuestro contenido multimedia. De este modo, un Period se compone de un tiempo de inicio, una duración determinada y uno o varios Adaption Sets (en la *Figura 10*, Group).

El tiempo de inicio se concreta de diferentes maneras. Como posteriormente se especificará, existen diferentes atributos dentro de cada MPD, para el tiempo de inicio se usará el atributo @start.

- **(3) Adaption Sets:**

Como anteriormente se ha recalcado, cada Period contiene uno o varios Adaption Sets. Un Adaption Sets ofrece información sobre uno o varios contenidos multimedia y sus diferentes alternativas de codificación. De este mismo modo, un Adaption Sets consta de uno o varios Represantions, donde cada uno será reproducido en cada momento concreto, por lo que se puede decir que contiene streams equivalentes.

- **(4) Representation:**

Cada Representation es una posible codificación de un contenido multimedia. Cada Representation se diferencia de otro por su campo Segment Info, que nos proporciona el bitrate, resolución, número de canales y otras posibilidades.

- **(5) Segments:**

Es una de las varias partes en las que se ha dividido el stream. Contiene una URI, gracias a la cual encontraremos dónde está situado el contenido que queremos reproducir mediante una petición HTTP GET. También proporciona información sobre la ubicación, la disponibilidad y las propiedades de los Segments de un Representation.

A continuación vamos a exponer los diferentes atributos de los que consta un fichero MPD. Los mostraremos en la siguiente tabla [8]:

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
@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 @		

Visto el formato de un fichero MPD, procedemos a mostrar un ejemplo de su estructura en XML en la que se muestra una parte del fichero de información que recibirá el cliente en su primera solicitud al servidor:

```
<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">
```

Se deja a continuación, en la siguiente URL, una página web donde podremos testear si nuestros ficheros MPD están formados correctamente.

http://www-itec.uni-klu.ac.at/dash/?page_id=605#

I.2 Formato de los segmentos.

Una vez conocemos la descripción del contenido disponible hay que adentrarse en el formato de los segmentos.

El formato de los segmentos especifica la semántica y la sintaxis de los recursos asociados con HTTP-URL en el MPD. DASH se basa en formatos de segmentos regidos por formatos de contenedores MPEG, MPEG-2 Transport Stream e ISO base media file format [10].

MPEG-2 TS, es un formato de contenedor que encapsula Packetized Elementary Streams (define la realización de flujos elementales, por lo general la salida de un codificador de audio o vídeo, en paquetes dentro del flujo de transporte MPEG) con sincronización de los streams y corrección de errores para cuando haya una degradación de la señal.

ISO base media file format, formato que define para archivos multimedia una estructura general según ISO/IEC 14496-12 (MPEG-4 Part12).

Por último, se define los cuatro tipos de segmentos existentes en DASH:

* ***Initialization Segment***. Contiene la información necesaria para la reproducción de los segmentos.
* ***Media Segment***. Contiene y encapsula los streams.
* ***Index Segment***. Contiene información de indexación para uno o varios Media Segments.
* ***Bitstream Switching Segment.*** Contiene información para cambiar a la Representation correspondiente.

I.3 libdash.

libdash es una librería interna que implementa el estándar completo MPEG-DASH de acuerdo con ISO/IEC 23009-1:2012. Proporciona un objeto orientado a la interfaz (OO) para el estándar MPEG-DASH basado en XML (MPD) y elementos multimedia (segmentos). Dicha librería proporciona interfaces externas para:

- *el streaming de control DASH*, es decir, la lógica de adaptación (planificación de descarga) que determina qué segmento se descarga será el siguiente basándose en un determinado contexto para el cliente;

- *el analizador de segmento*, es decir, para manejar segmentos descargados (ISO base media file format, MPEG-2 TS) y formatos de codificación encapsulados;

- *el reproductor multimedia* que es finalmente responsable de decodificar y hacer la presentación multimedia.

libdash proporciona los medios necesarios para acceder a la información dentro del MPD y poder programar la descarga de los elementos multimedia (segmentos) descritas en el mismo [38]. *libdash* deliberadamente no proporciona implementaciones para los módulos antes mencionados ya que pueden variar entre diferentes casos de uso y escenarios de implementación, y también puede ir más allá de lo normativamente definida en ISO/IEC23009-1:2012 (por ejemplo, incluir codecs). La motivación detrás de este enfoque es proporcionar a los desarrolladores una herramienta que apoya plenamente la normativa y permite la integración del mismo dentro de un ecosistema de medios de comunicación.

Para comprender el funcionamiento de esta librería vamos a expandir la explicación dada en la *Figura 8* sobre el funcionamiento entre un cliente y servidor.

En la *Figura 11*, las piezas de color naranja son los formatos MPD y segmentos estandarizados. La entrega del propio MPD, la heurística de control y el propio reproductor se representan en las figuras azules. *libdash* encapsula el MPD Parser (analizador MPD) y el HTTP Manager (gestor HTTP). Por lo tanto, la biblioteca ofrece interfaces para el Control de Streaming DASH y para el reproductor multimedia para acceder al MPD y a los segmentos multimedia descargables. El orden de descarga de

dichos segmentos no serán manejados por la biblioteca, de esto se deja al Control de Streaming DASH, que es un componente propio en esta arquitectura, pero que también podría ser incluida en el reproductor multimedia.

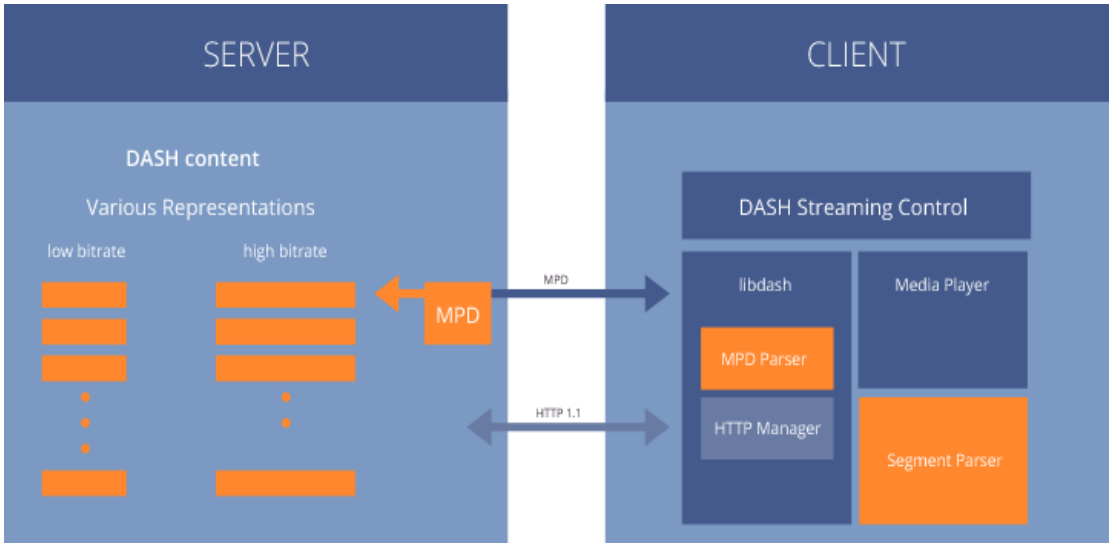


Figura 11 – libdash entre cliente y servidor DASH.

En una implementación típica, un servidor DASH ofrece segmentos en varios bitrates y resoluciones. El cliente recibe inicialmente el MPD a través del *libdash,* que proporciona una interfaz orientada a objetos simple para el manejo del MPD. Como ya bien sabemos, el MPD contiene, entre otras informaciones, las relaciones temporales para las diversas calidades y segmentos. Con esta información, el cliente puede descargar los segmentos multimedia individuales a través de *libdash* en cualquier momento. Las adaptaciones al ancho de banda y calidad de los segmentos no se realizarán a través de *libdash* ni del estándar MPEG-DASH sino que se dejará en manos de la aplicación que esté utilizando la librería.

II GPAC

GPAC es un framework multimedia de código abierto. Su función básicamente reside en fines académicos y de investigación. El proyecto se focaliza principalmente en tecnologías de animación y en formatos de encapsulación multimedia como MP4. GPAC es multiplataforma y está escrito en C por razones de portabilidad, tratando de mantener el consumo de memoria lo más bajo posible [15]. En la actualidad se ejecuta en Windows, Linux, MacOSX, Android, iOS (iPhone y iPad), WindowsMobile y algunos sistemas SymbianOS.

Este framework cubre diferentes aspectos multimedia como, codecs A/V, protocolos de red, algoritmos de gestión de sincronización, herramientas de representación, etc.

Como ya hemos visto en apartados anteriores, un ejemplo de cliente DASH viene dado por el proyecto GPAC. El cliente es capaz de cargar y reproducir un gran

subconjunto de la especificación DASH. El motor de acceso a DASH y los descargadores de HTTP se implementan como parte de *libgpac*, la biblioteca central de GPAC, y se pueden utilizar sin ningún tipo de dependencia con el reproductor multimedia. El cliente GPAC, normalmente llamado Osmo4 o MP4Client, es compatible con las siguientes características para DASH:

- Reproducción desde el disco duro local o servidores HTTP;
- La reproducción en directo y los perfiles simples con la adaptación de bitrate;
- Soporte para representaciones no bitstreamSwitching utilizando segmentos de inicialización separados;
- Soporte para el atributo @group y la de/selección automática de los conjuntos dentro de un grupo.
- La calidad puede ser cambiada manualmente, se puede configurar para cambiar todos los N segmentos o se puede configurar por el controlador de velocidad de adaptación interna.

GPAC proporciona, de esta manera, tres principales herramientas basadas en una librería propia llamada *libgpac*. Estas herramientas son:

- **Osmo4/ MP4Client.** Reproductor multimedia que soporta DASH. Como ya veremos para nuestro proyecto usaremos Osmo4 ya que dispondremos de él en formato *.apk* (aplicación para Android).
- **MP4Box.** Empaquetador multimedia que utilizaremos para que nuestros vídeos adquieran el formato establecido por el estándar DASH.

Con estas dos herramientas ya tendremos todo lo necesario para que cualquier cliente que así lo desee pueda reproducir vídeos basados en el estándar DASH.

II.1 MP4Box.

El empaquetador multimedia que se usa en GPAC es MP4Box, su utilidad se centra en la realización de numerosas manipulaciones con archivos multimedia. A nosotros, particularmente, nos interesa su capacidad para generar contenidos conformes a la especificación MPEG-DASH.

MP4Box NO realiza trascodificación de audio/vídeo/imagen (re-codificación de pistas multimedia a un formato codificado diferente). Para codificar el contenido, necesitará otras herramientas [16].

Las opciones que nos proporciona M4Box son las siguientes:

- **Intercalado.** Se produce cuando muestras de diferentes pistas se almacenan alternativamente en el archivo, por ejemplo: N milisegundos de muestras de vídeo, seguido de N milisegundos de muestras de audio, seguido de N milisegundos de muestras de vídeo, etc. Típicamente, las muestras intercaladas son agrupadas dentro de una ventana de entrelazado. El

intercalado reduce los accesos al disco, los requisitos de buffer de reproducción y permite la descarga progresiva y la reproducción.

- **Fragmentación.** Es un proceso opcional aplicable al formato MP4. Predeterminadamente, los archivos MP4 generados con MP4Box no están fragmentados. Este proceso consiste en el uso de Movie Fragments (moof), herramienta introducida en el ISO para mejorar la grabación de largas secuencias y que actualmente se usa para la transmisión HTTP.
- **Segmentación.** Proceso de creación de segmentos, las partes de un archivo original dan lugar a la descarga HTTP individual/separada. Un segmento es a lo que se refiere el archivo XML usado para conducir el HTTP Streaming.
- **Split.** MP4Box puede por último, dividir un archivo y crear archivos reproducibles individuales del original. No utiliza segmentación, se elimina la fragmentación y puede utilizar intercalado.

En resumen, MP4Box se puede utilizar:

- Para manipular archivos ISO como MP4, 3GP: agregar, quitar la multiplexación de datos de audio, video y presentación (incluyendo subtítulos) de diferentes fuentes y en diferentes formatos.
- Para los idiomas de presentación de codificación/decodificación como MPEG-4 XMT o W3C SVG a/desde formatos binarios como MPEG-4BIFS o LASeR.
- Para realizar el cifrado de flujos.
- Para la fijación de los metadatos para flujos individuales.
- Para preparar los archivos para diferentes protocolos de entrega, descarga o streaming HTTP principalmente RTP (lo que más nos interesa).
- Y el envasado y etiquetado de los resultados para el streaming, descargar y reproducir en diferentes dispositivos (por ejemplo, teléfonos, PDA) o un software diferente (por ejemplo iTunes).

Es ampliamente utilizado: por académicos, video community y se utiliza en muchas infraestructuras de nube para la preparación de archivos multimedia para la reproducción, en especial para iTunes/iOS y PlayStation.

Un ejemplo de cómo se utilizaría MP4Box para crear un archivo multimedia según el estándar DASH sería la siguiente línea de comandos, modificando la calidad según nuestros criterios:

```
/Applications/Osmo4.app/Contents/MacOS/MP4Box -dash 5000 -frag 1000 -subsegs-per-sidx 5 -url-template -segment-name %s_dash -base-url http://163.117.166.10/sintel -out sintel_dash.mpd sintel_3000kbps.mp4
```

II.2 Osmo4.

GPAC proporciona un reproductor multimedia altamente configurable disponible en muchas plataformas. Este reproductor es más que un reproductor audiovisual tradicional porque, además de su capacidad para reproducir gran parte de

los formatos de vídeos o audio y su apoyo a la mayoría de los protocolos de entrega existentes, se centra en los gráficos, tecnologías de interactividad y las animaciones.

Antes de adentrarnos más profundamente en Osmo4, hay que recalcar la compatibilidad con el estándar DASH, ya que es el principal motivo por el que se usa en este proyecto. Osmo4 se basa en muchos paquetes de código abierto y se utiliza para mostrar contenido 2D y audiovisual [17].

Como bien sabemos ya, Osmo4 es compatible con una buena parte de la tecnología DASH. Los principales puntos en común son los siguientes:

- Puede reproducir desde un servidor HTTP o desde el almacenamiento local para fines de prueba.
- Los segmentos multimedia están basados en el TS e ISOBMF.
- Utiliza descarga de componentes independientes (un conjunto de adaptación para audio, y una para vídeo)
- Soporta sintaxis MPD (muy útil para nuestro proyecto).
- Posibilidad de cambio manual de la calidad.
- Cambio de la calidad básica automáticamente cuando se utilizan HTTP-URLs.

III Redes CCN

Las redes de datos se crearon principalmente para resolver el problema de la distribución de los recursos. El modelo de comunicación resultante se convirtió en una comunicación entre dos máquinas, con una máquina solicitante de datos y otra que daba acceso a los datos. En la arquitectura IP, se realiza la comunicación entre hosts identificados por direcciones y sobre una asignación permanente de contenidos para albergar lugares. Como ya hemos visto, el contenido de datos está aumentando actualmente y para este tipo de redes de datos de crecimiento, es más conveniente centrarse en los datos en sí que en lugar de ubicación de los mismos. Para ello se han propuesto iniciativas de redes centradas en la información, como las Content Centric Networking (CCN) [30].

III.1 Introducción.

Content Centric Networking (CCN) es una arquitectura de comunicación basada en datos con nombre, donde la identificación y el transporte de contenidos dependen de sus nombres y no de su ubicación. El enfoque CCN aborda las cuestiones que limitan el uso actual de las redes mediante el aumento de la disponibilidad de datos. Proporciona almacenamiento en caché de los propios routers, para reducir la congestión y mejorar la velocidad de entrega. En términos de seguridad, CCN sugiere que la confianza en el contenido reside en la ubicación de los datos. Por ello, se construye la seguridad en la red a nivel de datos. Además, debido a que la comunicación se basa sólo en nombres de datos, no se realiza ninguna asignación entre los contenidos y ubicaciones [11]. Por lo tanto, la configuración de los dispositivos de red es mucho más simple.

III.2 Protocolo CCNx.

Dentro de esta red tenemos una implementación del protocolo que hemos usado para este proyecto debido a los múltiples beneficios que nos aporta. Esta implementación del protocolo se llama CCNx y es una serie de APIs y aplicaciones para la arquitectura de comunicación Content-Centric Networking. De acuerdo con las especificaciones de CCN, que se basa en los datos de nombre, donde el nombre del contenido sustituye a la dirección de ubicación; el protocolo CCNx ofrece servicios de entrega independientes de la ubicación de los paquetes de datos con nombre [31]. Los servicios incluyen el reenvío de saltos múltiples para la entrega de extremo a extremo, control de flujo, la entrega multidifusión transparente y automática con almacenamiento de reserva disponible en la red, el reenvío de trayectos múltiples sin bucles y la verificación de la integridad del contenido, independientemente de la ruta de entrega.

Aunque el protocolo CCNx está diseñado para ofrecer contenidos basados en sus nombres, las aplicaciones se pueden ejecutar en la parte superior de la UDP o TCP para tomar ventaja de la conectividad IP existente. Dado que el contenido se denomina independientemente de la situación en el protocolo CCNx, también se puede conservar indefinidamente en la red. Cada paquete de datos puede tener una caché en cualquier enrutador CCNx. Gracias a esta caché, el uso de la red es más eficiente cuando muchas personas están interesadas en el mismo contenido ya que se proveerán del contenido multimedia en el primer enrutador CCNx, puesto que ya lo tendrá almacenado.

El protocolo CCNx es compatible con una amplia gama de aplicaciones de red dejando la elección de los convenientes nombres a la aplicación. Puede ser natural pensar en las aplicaciones de contenidos almacenados, pero el modelo CCNx también es compatible con la comunicación en tiempo real (que es lo que más nos interesará para nuestro proyecto) y los protocolos de detección, y es lo suficientemente genérico como para llevar a conversaciones entre hosts, como conexiones TCP.

La flexibilidad del protocolo CCNx para ser desplegado en diferentes ambientes hace que tomemos la iniciativa de aplicar la arquitectura Content-Centric Networking en sistemas embebidos, y en particular las redes que usan contenido streaming.

III.3 Diseño CCN.

La idea básica de la creación de redes centradas en el contenido es, como se dijo antes, hacer del elemento central el contenido de las operaciones de red. En consecuencia, la arquitectura se diferencia del modelo común de capas ISO/OSI, como se representa en la *Figura 12*.

Según la nueva arquitectura de que propone CCN, la actual pila de protocolos está definida por un modelo de “reloj de arena”, donde IP ha servido de nexo de unión entre todas las tecnologías que hay por encima y por debajo de esta capa, y ha permitido su evolución. Gracias a esto se ha obtenido una mayor interoperabilidad y

estabilidad en entornos heterogéneos, pero también ha reducido la posibilidad de evolución de la capa de red. En esta nueva propuesta se desplaza la parte más estrecha del modelo hacia arriba y deja que el único nexo de unión entre las diferentes tecnologías sean los datos.

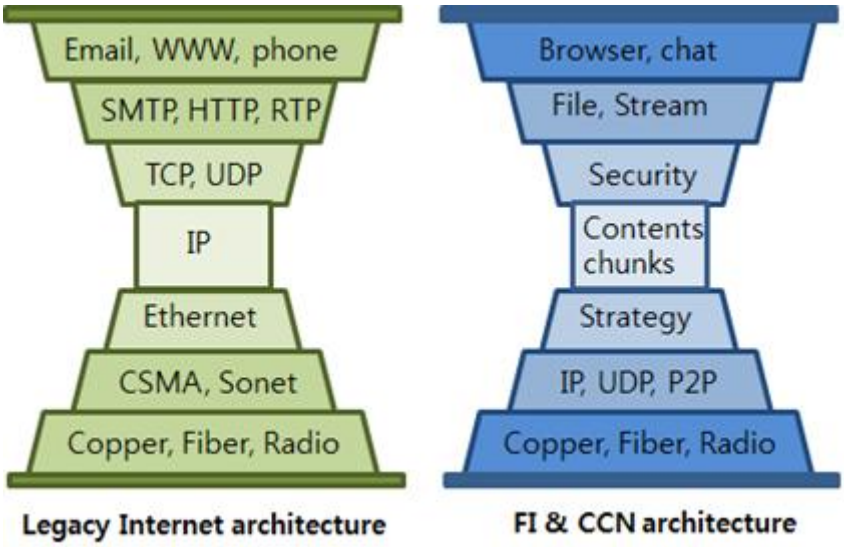


Figura 12 – Cambio del cuello de botella de la pila de comunicación al contenido chunk.

El contenido se convierte en el nexo entre todos los participantes de la red. Por lo tanto, en la nueva capa, el contenido de la fuente deseada no es relevante [33]. Además, también hay una capa de estrategia y una capa de seguridad introducida por debajo y encima de la capa de contenido para mejorar esta pila. Cabe destacar que el contenido es en gran medida independiente del tipo de conexión que se sirve. Por lo tanto, soporta de forma nativa multihoming, así como tolerancia con las interrupciones de la conectividad.

III.4 Tipos de paquetes.

En el modelo CCN existen dos tipos de paquetes:

- **Interest packet.** Es una solicitud de datos con nombre. Contiene el nombre completo que identifica un fragmento de contenido. Este contenido se recuperará específicamente a su disposición en un nodo de la red. También puede contener sólo un prefijo del nombre de contenido (content name). Entonces, cualquier contenido cuyo nombre coincide con el nombre del prefijo de interés puede ser una posible respuesta a este interés. Además, se acompaña con la información de selección (selector), tales como el alcance de la red, donde los datos deben provenir de cierta información o filtro. Por último, contiene un nonce, utilizado para detectar los intereses duplicados. A continuación se muestra el formato de estos paquetes:

Content Name
Selector
Nonce

- **Data packets.** Este tipo de mensajes se utilizan para suministrar los datos. Contiene una carga útil de datos precedido por el nombre de identificación. Los Data packets se dicen que satisfacen a los Interest packet y mantienen una relación uno-a-uno, donde los data consumen Interest packet. Esta regla mantiene un equilibrio de flujo en cada salto y evita la congestión en el centro de una ruta de conexión. Los nombres en las redes CCN son jerárquicos, con la consecuencia, que los Data packets sólo sirven un interés packet, si su prefix-name coincide con el nombre de los Interest. Además del nombre y datos binarios arbitrarios también contiene una firma digital de algunos resúmenes criptográficos del paquete, así como otra información firmada. El último campo mencionado proporciona información adicional sobre el paquete como el ID del editor, dónde localizar la clave para comprobar la firma o un sello de tiempo. Para servir a estos "nombres activos" los editores pueden generar contenido dinámicamente para satisfacer las exigencias de la moderna Internet. El formato de los paquetes es el siguiente:

Content Name
Signature
Signed Info
Data

III.5 Modelo de los nodos.

De acuerdo con las especificaciones del contenido CCN, un nodo requiere las siguientes estructuras de datos para proporcionar almacenamiento en búfer/caché de datos, gestionar las solicitudes de contenido y reenviar mensajes a otros nodos de la red.

Un CCN realiza operaciones similares en comparación con los de un router IP normal. Los paquetes que se entregaron en una face (interfaz) se someten a longest prefix matching en su campo de nombre y se procesan de acuerdo a la información almacenada en las tres estructuras de datos principales que se mantienen en cada nodo:

- **Content Store (CS).** Es una caché donde se almacenan los datos. Dado que el contenido es de auto-autenticación y auto-identificación de cada paquete, pueden ser útiles para cualquier participante potencial en la red cercana. La capacidad de servir contenido directamente en lugar de generar nuevas búsquedas minimiza el uso del ancho de banda y la latencia. En comparación con un router IP, que utiliza la estrategia de sustitución de la memoria intermedia, se sustituye dicha estrategia por una estrategia utilizada menos recientemente (LRU) o con menos frecuencia (LFU) para aumentar la probabilidad de un acierto de caché.
- **The pending interest table (PIT).** El PIT es una tabla de fuentes de Interest pendientes de ser satisfechas. Cada entrada en el PIT puede apuntar una lista de identificadores de las face (interfaces) que son las fuentes del Interest

insatisfecho. Debido a que los Interest son los únicos paquetes enrutados en CCN hacia la fuente de los contenidos, tan pronto como el Interest llega a una fuente de contenido, el PIT sirve como marca en el rastro de los datos hacia su(s) solicitante(s). Por estos medios los data satisfacen los Interest y se retira la entrada en el PIT.

- **The forwarding information base (FIB).** Es una tabla de destinos para los Interest, organizados para su recuperación en la consulta de coincidencia de prefijo en los nombres. Una entrada en el FIB puede ser un prefijo que apunta a una serie de destinos en lugar de una específica. FIB juega un papel equivalente a la tabla de enrutamiento IP, ya que indica por qué interfaz debe un nodo enviar su mensaje de Interest para recuperar un contenido coincidente. Una entrada en una tabla FIB contiene un prefijo y una lista de identificadores de faces (interfaces). Para recuperar los datos que coincida con el prefijo, un nodo tiene que enviar un Interest a través de, al menos, una de las interfaces cuyo identificador esté incluido en la lista. En este trabajo, ningún mecanismo o protocolo está implementado para llenar las tablas FIB dinámicamente. Se hace estáticamente durante la inicialización de la aplicación que se ejecuta en la parte superior de un nodo antes de iniciar la comunicación con los otros nodos en la red. Así que las entradas en la FIB no cambiarían o expirarían durante la ejecución de la aplicación.

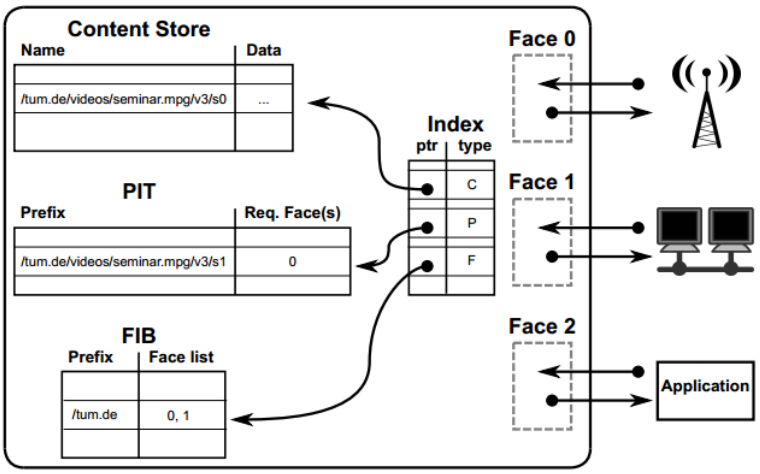


Figura 13 – Modelo de un nodo CCN.

En la *Figura 13* se muestra como interactúa cada estructura de datos en un nodo CCN según lo especificado y definido anteriormente.

III.6 Nomenclatura y transporte.

El esquema de nombres de CCN está diseñado para tener un formato legible por la mayoría de personas. Al mismo tiempo que trata de cumplir con las demandas de enrutamiento, es decir, las capacidades de enrutamiento eficientes sobre nombres, y se inspira en los mecanismos de funcionamiento de TCP [32]. Por lo tanto, como se puede ver en la *Figura 14*, los nombres están jerárquicamente estructurados y formados por varias componentes.

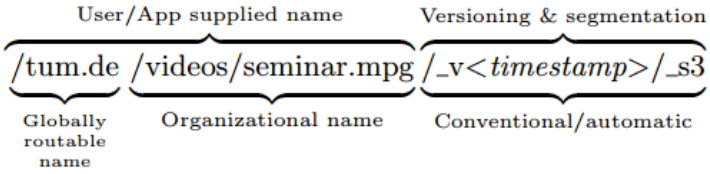


Figura 14 – Ejemplo legible de un data name.

Por conveniencia notacional, la representación legible para humanos es elegido por los diseñadores. Se divide en un nombre proporcionado por el usuario o una aplicación, que consiste en un nombre globalmente enrutable y el nombre del contenido dentro de la estructura organizativa de su origen. En segundo lugar, la cola del nombre, que se supone que es una convención de nomenclatura estándar que refleja la versión de un determinado contenido y su segmentación. El esquema permite tener un orden total en el contenido que se puede reflejar en un árbol como en la *Figura 14*. El acceso al contenido significa el recorrido de dicho árbol. Una característica incremental, obtenida mediante la imposición de un orden total, es la capacidad de dirección de contenido en relación con información conocida.

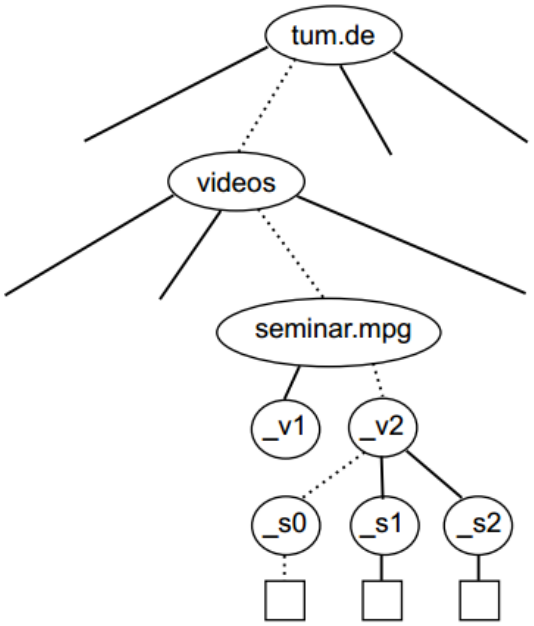


Figura 15 – Árbol trasversal de nombres.

III.7 Algoritmos de comunicación en CCNx.

Processing Interest Messages.

Un mensaje Interest recibido se procesa de acuerdo a los siguientes pasos:

* Operaciones de búsqueda en el Content Store. Si el CS tiene un Content Object (Data packet) cuyo nombre coincide con el prefijo del mensaje Interest, transmitirá el Content Object coincidente a la interfaz donde llegó el Interest. Si el CS tiene múltiples Content Object que coincidan al mismo tiempo, el

primer Content Object enviará su respuesta al mensaje Interest. Cuando se encuentra una coincidencia en la CS, el proceso se detiene y el mensaje de Interest se descarta, ya que ha sido satisfecho.

* Operaciones de búsqueda en el PIT. Si un mensaje de Interest coincidente se encuentra en el PIT, significa que un mensaje de Interest equivalente ya se ha enviado y está en espera. Si hay una entrada PIT de coincidencia exacta, la interfaz de llegada del Interest se añadirá a la lista de la interfaz de entrada PIT y el Interest será descartado.

* Operaciones de búsqueda en el FIB. Si un prefijo coincidente se encuentra en la FIB, se crea una entrada en el PIT del mensaje Interest y su interfaz de llegada, y el mensaje se transmitirá a las interfaces registradas de destino para el prefijo en el FIB, excepto en la interfaz de llegada si está registrado a la entrada FIB coincidente.

* Si no se encuentran coincidencias en los pasos anteriores entonces el nodo no tiene manera de satisfacer el mensaje de Interest y se desprenderá de ella.

A continuación se deja en la siguiente figura [*Figura 16*], un esquema del procesamiento de los mensajes Interest para una comprensión más sencilla. En la *Figura 17* se presenta el procesamiento de mensajes de contenido, que se producen cuando se ha realizado ya alguna petición entrante. Ya que es muy similar al anterior caso sólo vamos a exponer el esquema para facilitar la comprensión del mismo.

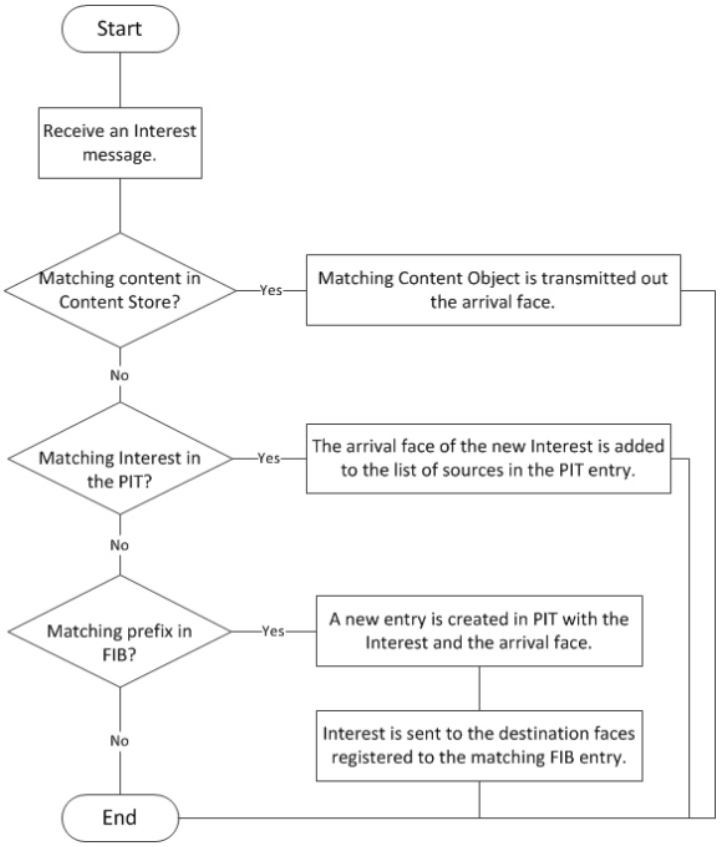


Figura 16 – Procesamiento de Interest entrante.

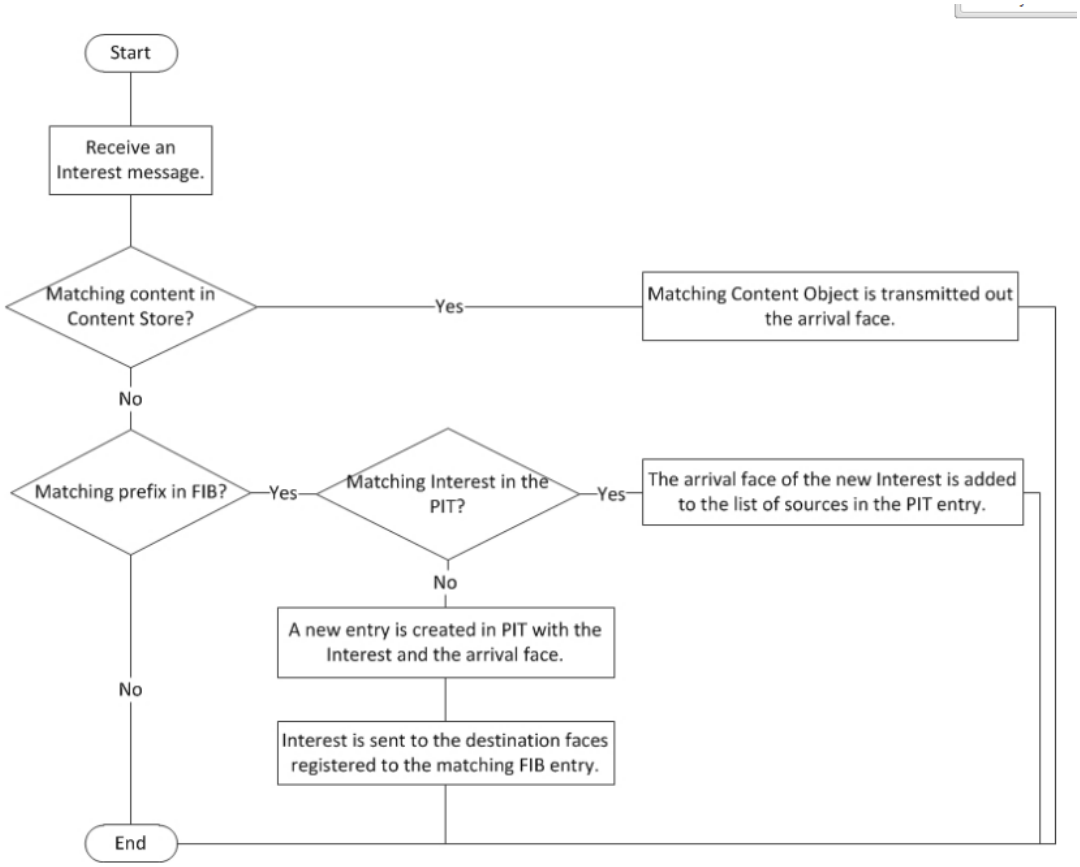


Figura 17 – Algoritmo de modificación para el procesamiento de Interest entrantes.

Parte 3
Diseño e implementación

En esta parte del documento se hablará de las partes más prácticas, el diseño y la implementación. En esta sección se verá más ampliamente el entorno en el que se ha desarrollado nuestro estudio y la tecnología implementada para su usabilidad.

En el diseño se describirá el proceso de definición de la arquitectura, componentes y otras características del sistema que nos ayudarán a visualizar más tarde el tipo de baterías de prueba realizadas. En cuanto a la implementación, se hablará de cómo se ha realizado toda la estructuración y los problemas que fueron apareciendo.

I Diseño

En primer lugar se introduce el acoplamiento del estándar DASH a las redes CCN para posteriormente poder adentrarse con una mayor soltura en el diseño que hemos seguido en este Trabajo Fin de Grado.

I.1 DASH sobre CCN.

Introducción.

Como ya se ha comentado en gran parte del proyecto, es un reto transmitir vídeo a través de la arquitectura actual de Internet, ya que este servicio es sensible a la latencia y ancho de banda. En primer lugar, los nodos intermedios, tales como routers IP están diseñados para descartar los paquetes después de que hayan sido enviados. Esto conduce a una pérdida de ancho de banda de la red y la congestión de la misma cuando se solicita el contenido en un mismo intervalo. En segundo lugar, como IP proporciona una entrega de mejor esfuerzo es probable que disminuya la probabilidad de entrega con éxito cuando la distancia entre el usuario y el servidor de contenidos es considerablemente amplia. Por ello se han propuesto varias soluciones como redes Peer-to-Peer (P2P) y Content Delivery Network (CDN) que intentan paliar estos problemas.

P2P se caracteriza porque es una red de computadoras donde se intenta funcionar sin clientes ni servidores, sino que una serie de nodos que se comportan por igual entre ellos mismos [34]. Es decir, actúan tanto de servidores como de clientes, esto quiere decir que cada nodo tiene almacenada cierta información que puede compartir con los demás nodos. Pero tiene también algunos aspectos negativos que han hecho que se busquen otras alternativas. Estas desventajas son: a medida de que la red crece se vuelve más difícil de coordinar y operar, falta de seguridad y que el sistema no es centralizado y esto dificulta la administración.

CDN nos proporciona otra variante que también intenta corregir los problemas de los que se ha hablado. CDN es una red superpuesta de computadoras que contienen copias de datos de los servidores colocados estratégicamente en varios puntos de la red para que el cliente acceda en un tiempo menor a la información requerida [35]. El problema de esta solución es que es relativamente compleja de aplicar y su gestión es costosa. Por otro lado, CDN es escalable pero no es fácil de lograr una rápida expansión de la red debido a los costos administrativos y de operación.

Una vez vistas estas arquitecturas se va a proceder a explicar la arquitectura que se ha desarrollado en nuestro estudio. En esta parte, por tanto, se procede a presentar el sistema de DASH sobre redes CCN, en adelante DASC [5].

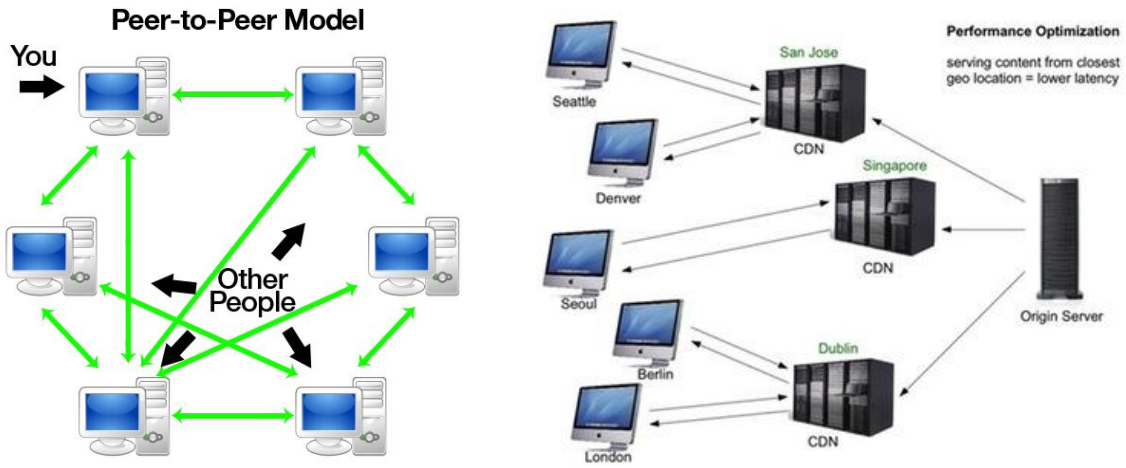


Figura 18 – Arquitectura P2P y CDN.

Aunque DASH está diseñado principalmente para HTTP, su diseño modular permite HTTP para ser sustituido por nombres de contenido CCN representados como identificadores uniformes de recursos (URI), mientras que HTTP utiliza Localizadores Uniformes de Recursos (URL). El esquema de nombres del CCN-URI para segmentos DASC es compatible y refleja simplemente la HTTP-URL de un segmento DASH con prefijo *ccnx:/DASC* donde *ccnx:* identifica el esquema de CCN-URI y DASC indica un segmento DASC (que también está disponible por HTTP en caso de que el segmento no está disponible dentro de la red CCN) [14]. A continuación se muestra los dos tipos de formatos, aunque en nuestro caso el segundo formato no se ha utilizado:

El nombre para un segmento DASH:
http://www-itec.uni-klu.ac.at/BigBuckBunny/bunny_2s/bunny_2s_150kbit/bunny_2s1.m4s

El nombre para un segmento DASC:
ccnx:/DASC/http/www-itec.uni-klu.ac.at/BigBuckBunny/bunny_2s/bunny_2s_150kbit/bunny_2s1.m4s

Hay que dejar claro que en el enfoque CCN, existen sólo dos tipos de paquetes: de interés y de datos. Los primeros se utilizan para solicitar el contenido, mientras que los segundos se utilizan para su entrega efectiva. La carga útil máxima de un paquete de datos es de 4096 bytes y también se hace referencia como un CCN chunk. Los paquetes de datos se manejan de manera eficiente en los nodos de la red, por ejemplo, para satisfacer a los paquetes de interés consolidados procedentes de múltiples clientes y, por lo tanto, proporcionar apoyo implícito para la multidifusión y el almacenamiento en caché de paquetes de datos en los nodos de CCN dentro de la red de entrega [13]. Por todo esto cabe destacar que los nodos de CCN enviarán el paquete interés basado en un esquema de emparejamiento de longest-prefix-match que también se utiliza en el enrutamiento IP.

Integración de DASH en CCN.

El concepto básico de DASH es el uso de segmentos de contenidos multimedia, que pueden ser codificados en diferentes tasas de bits, resoluciones, etc. llamados representations. Estos segmentos son servidos por los servidores Web HTTP

convencionales y pueden ser abordados a través de solicitudes HTTP GET del cliente. Como consecuencia, el sistema de streaming está basado en la demanda, y toda la lógica de la transmisión se encuentra en el cliente, lo que permite adaptar el flujo multimedia a sus capacidades. Además de esto, el contenido puede ser distribuido utilizando CDNs convencionales y su infraestructura HTTP, que también escala de manera eficientemente. Con el fin de especificar la relación entre los segmentos de contenido multimedia y la tasa de bits asociada, la resolución y la línea de tiempo, se utiliza los Media Presentation Description (MPD).

En contraste a la red CCN, la tecnología detrás de DASH ya está muy avanzada y están disponibles por empresas de la industria más importantes (por ejemplo, Adobe, Apple, Microsoft). Como ya hemos comentado [2.I], DASH tiene por objeto permitir el streaming adaptativo, es decir, cada fragmento se puede proporcionar en diferentes calidades, formatos, idiomas, etc. para hacer frente y poder adecuarse a la diversidad de redes y dispositivos de hoy en día. Como este es un requisito importante para futuras propuestas de Internet como CCN, la combinación de esas dos tecnologías parece ser obvia.

Dado que estas dos propuestas se encuentran en diferentes capas de protocolo (DASH en la aplicación y CCN en la capa de red) se pueden combinar de manera muy eficiente para aprovechar las ventajas de ambos y potencialmente eliminar las desventajas existentes. En principio, hay dos opciones para integrar DASH y CCN: un servicio de proxy actúa como un intermediario entre HTTP y CCN, y que el cliente DASH implemente una interfaz nativa de la CCN [12]. El primero, transforma una solicitud HTTP a un paquete de interés correspondiente, así como un paquete de datos a una respuesta HTTP, incluyendo el transporte fiable como los ofrecidos por TCP. Esto puede ser un buen compromiso para implementar CCN en una red administrada. Como proxy nos estamos centrando en un enfoque más integrado, con el objetivo de aprovechar al máximo el potencial de la red CCN. Por otro lado, otra posibilidad es una interfaz nativa CCN dentro del cliente DASH, que adopta un esquema de nombres CCN (CCN URI) para denotar los segmentos en los MPD.

La *Figura 19* presenta la arquitectura propuesta de DASH sobre CCN, donde los componentes relacionados con DASH están marcados en rojo, los componentes relacionados con CCN en verde, y algunos componentes de implementación en azul. Como se puede ver, sólo el componente de acceso a la red en el cliente tiene que ser modificado y los segmentos URI dentro del MPD deberán actualizarse de acuerdo con el esquema de nombres CCN.

Inicialmente, el cliente DASH recupera el MPD que contiene la CCN URI de las Content Descriptions, incluidos los segmentos multimedia. En esta primera fase hay que recalcar que en esta petición que realiza el cliente del MPD no se utiliza en ningún momento la red CCN, sino que es una comunicación directa entre el cliente y el servidor. El esquema de nombramiento de los segmentos puede reflejar características intrínsecas del CCN como versiones y soporte de segmentación como se muestra en la *Figura 20*. Este soporte de segmentación es obligatorio para el streaming multimedia en CCN y, por lo tanto, también se puede aprovechar para la transmisión basada en

DASH sobre CCN. Las versiones de CCN pueden ser adoptadas para señalar las diferentes representaciones del contenido basado en DASH, que permite una adaptación implícita del contenido solicitado a las condiciones de ancho de banda de los clientes. Es decir, el paquete de interés ya proporciona las características deseadas de un segmento (por ejemplo, tasa de bits, resolución, etc.) dentro del nombre del contenido. Además, si las condiciones de ancho de banda de las interfaces correspondientes o rutas de enrutamiento lo permiten, los segmentos multimedia DASH podrían ser agregados automáticamente por los nodos de CCN, lo que reduce la cantidad de paquetes de interés necesarias para solicitar el contenido.

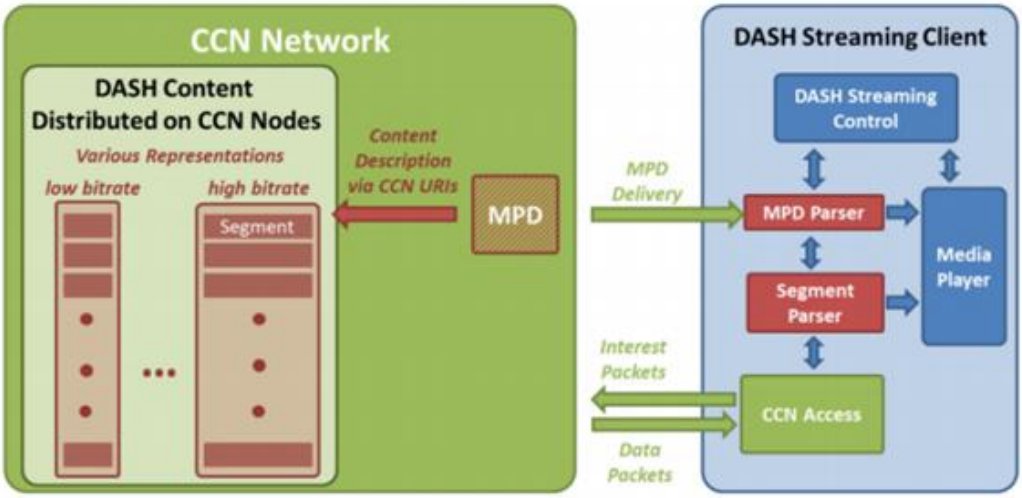


Figura 19 – Arquitectura DASH sobre CCN.

Después de solicitar el MPD, el cliente DASH comenzará a solicitar segmentos particulares. Por lo tanto, los paquetes de interés CCN son generados por el componente de acceso CCN y remitidas a las interfaces disponibles. Dentro del CCN, estos paquetes de interés se aprovechan para la agregación eficiente de interés para, por ejemplo, el contenido popular, así como la compatibilidad con multidifusión implícita. Por último, los paquetes de interés son satisfechos por los paquetes de datos correspondientes que contienen los datos del segmento de vídeo, que se almacenan en el servidor de origen o en cualquier nodo CCN, respectivamente. Con una creciente popularidad del contenido, que se distribuirá a través de la red que provocó retrasos de transmisión más bajos y reduce los requisitos de ancho de banda para los servidores de origen y de los proveedores de contenido, respectivamente.

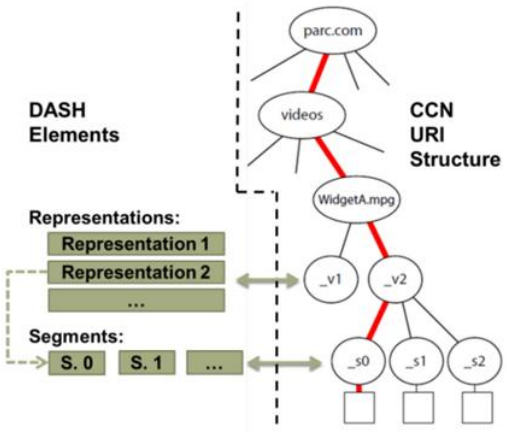


Figura 20 – Estructura de nomenclatura en CCN.

I.2 Diseño de arquitectura.

Para la realización del proyecto nos hemos basado en una arquitectura básica de una red CCN. Por ello, se va a mostrar un ejemplo de una arquitectura básica que nos ayudará a comprender mejor la integración de DASH y CCN.

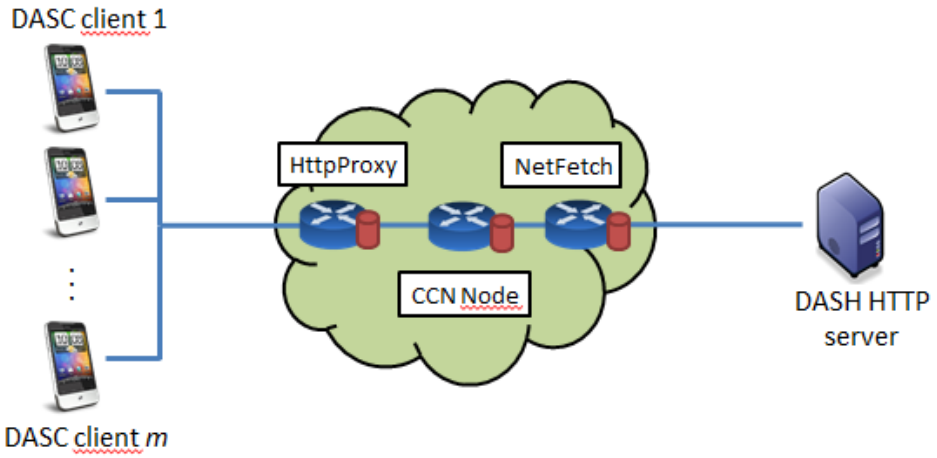


Figura 22 – Arquitectura red CCN simple.

La red local presentada a la izquierda es un despliegue de una red CCN [*Figura 22*]. El componente HttpProxy, que es proporcionado por CCNx, escucha los mensajes HTTP/IP y los convierte en mensajes CCN. Mientras que el componente NetFetch escucha los mensajes CCN y los convierte en mensajes HTTP GET. Una vez que el segmento de vídeo solicitado es recuperado a través de Internet a través de HTTP, NetFetch empaqueta en un objeto de contenido CCN que satisfaga el interés pendiente. El objeto de contenido (es decir, el segmento de vídeo) se almacena en caché en la red CCN para que una solicitud posterior para el mismo segmento de vídeo puede ser satisfecha inmediatamente.

Como ya se ha comentado anteriormente, la estructura de nombres utilizada es la siguiente:

http:/httpProxy/nombreSegmento

El mensaje de petición se dirigirá al HttpProxy, que lo convertirá en un Interest con nombre *nombreSegmento*. Si ese nombre se encuentra en la red CCN, se servirá de vuelta hasta el HttpProxy que lo volverá a construir como una respuesta HTTP. Si no está en la red CCN, llegará al NetFetch, que lo volverá a pasar a un GET de HTTP para enviarlo al nombre de segmento, que será el del servidor HTTP. Por ejemplo:

http://163.117.166.6:8080/hydra.netcom.it.uc3m.es/sintel-5s/sintel-256kbps_dash1.m4s

Profundicemos un poco más en esta red. Supongamos que un cliente quiere reproducir un vídeo streaming almacenado en un servidor web. Para ello se siguen los siguientes pasos:

1. El primer paso que deberá seguir el cliente será contactar con dicho servidor para obtener el MPD mediante una petición HTTP GET.

2. Una vez el cliente disponga ya del archivo MPD, podrá contrastar y calcular, mediante diferentes algoritmos que la propia aplicación de reproducción posee, que calidades tiene disponible, tasa de bits asociada, número de segmentos, proxy con quien comenzar la petición, etc.

3. Una vez que tiene toda esta información, el cliente se dispone a contactar con el proxy demandando el primer segmento de vídeo mediante una petición HTTP GET. El proxy (HttpProxy), es un proxy HTTP especializado que maneja un subconjunto de solicitudes HTTP GET y los convierte en paquetes Interest para que puedan ser tratados en la red CCN. Este proxy tiene la lógica suficiente para servir como un sustituto de los navegadores con proxies configurables.

4. Una vez realizada la conversión del mensaje HTTP GET a un Interest, se procede a recorrer la red CCN hasta llegar al NetFetch que nos servirá para revertir la conversión y que así el servidor web entienda la petición de streaming multimedia solicitado. NetFetch escucha los Interest de la red CCN y los convierte en peticiones HTTP GET. Una vez llega la petición al servidor web, éste comienza a servir los diferentes segmentos de vídeo que contiene ese streaming en concreto y los devuelve al NetFetch para que se los haga llegar al cliente.

5. Una vez llega al NetFetch el nuevo segmento, lo reenvía por la red CCN y almacenará este archivo en su caché interna. Gracias al NetFetch se asegura que NO se eliminen los archivos que pasan por la red CCN. Una vez realizado esto, el NetFetch manda el segmento al proxy para que pueda hacérselo llegar al cliente. Una vez llega a éste, comienza la reproducción del mismo. (Estos pasos se repetirían para los demás segmentos hasta completar en su totalidad todo el vídeo demandado).

6. Si otro cliente (contactaría con el servidor para obtener el MPD), o incluso el mismo, deseará reproducir el mismo el vídeo, una vez llegase a la red CCN, como ya están almacenados en caché dichos segmentos se reenviarían al cliente sin tener que pasar por el servidor (el gran beneficio de usar estas redes).

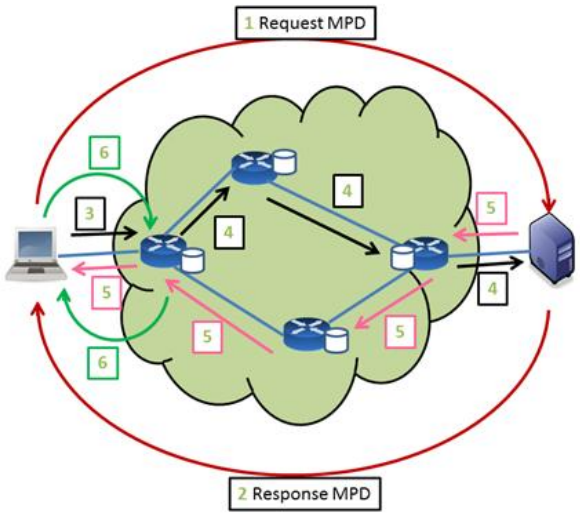


Figura 23 – Modelo de intercambio cliente-servidor DASH.

II Implementación

En esta sección se va a explicar más detalladamente cada parte de nuestra arquitectura. Como se ha visto en el apartado anterior [3.I.2] nuestro modelo tiene varias partes bien diferenciadas. Para entender como se ha configurado cada nuestra arquitectura se va a proceder a explicar cada una de las partes involucradas.

II.1 Cliente Android.

Como ya se ha ido comentado durante todo este documento, se ha focalizado el proyecto en la realización de la integración de un dispositivo Android a una red CCN sobre el estándar DASH y el posterior estudio de sus beneficios y desventajas.

Para el cliente Android se ha elegido un terminal HTC Legend con una versión de Android compatible [7.I] a la versión de Osmo4 (reproductor DASH utilizado) [2.II.2] 0.5.1-DEV-r4958. Como ya se ha hablado, gracias a este reproductor podemos reproducir los vídeos streaming con formato DASH. Para la configuración e instalación de este reproductor se han seguido diferentes pasos proporcionados en la página oficial de GPAC. Con estos pasos conseguimos poder desarrollar, cambiar y mejorar la aplicación Android para poder adaptarla a nuestras necesidades, ya que no nos descargamos el fichero .apk (paquete de instalación de aplicaciones Android), sino que accedemos a la configuración de la aplicación, el código.

Gracias a las herramientas que nos proporciona GPAC, se consigue tener de una manera muy bien distribuida el proyecto de la aplicación, junto a todas las librerías necesarias. A continuación se presentan los pasos que se siguieron para obtener este proyecto [18]:

1) El primer paso fue la descarga del directorio GPAC donde se trabajó mediante el repositorio SVN (referenciado como <GPAC>):

   svn co https://gpac.svn.sourceforge.net/svnroot/gpac/trunk/

2) Obtener los softwares de Android:

   A- Descargar el SDK de Android y descomprimirlo en cualquier ruta (referenciado como <SDK>).

   B- Descargar el NDK de Android y descomprimirlo (referenciado como <NDK>).

   C- En este paso se siguió primero la decisión de continuar todo el proceso desde el Shell. Debido a varios errores que fueron surgiendo se prefirió optar por la otra opción, utilizar Eclipse, que nos ayudaría más fácilmente luego para la simulación y compilación. Por lo que se mostrará únicamente los pasos seguidos con Eclipse. Una vez se tiene Eclipse se conseguirá el plugin ADT (Android Development Tools) para permitir a Eclipse interactuar con las herramientas de Android.

D- Obtener e instalar componentes SDK para Android. Se pondrá la ruta del directorio SDK (*Window->Preferences->Android*). A continuación, se obtendrán los paquetes necesarios para el correcto funcionamiento (*Window->AndroidSDK*) [Anexo I].

3) Compilar librerías adicionales para Android:

A- Construir todas las extra-libs. Ir a <GPAC>/trunk/gpac_extra_libs. Descomprimir el paquete.

Ir a <GPAC>/trunk/gpac_extra_libs/build/android. Dar permisos al ejecutar.

Ejecutar el siguiente comando: /gpac_build_all_extra_libs <NDK>

B- Copiar todas las extra-lib compiladas a la carpeta correcta <GPAC_DIR>/trunk/gpac/extra_lib/lib/android.

4) Construir GPAC y Osmo4.apk para Android:

Ir a GPAC_DIR/build/android/jni y lanzar el script ./gpac_build_android <NDK> <SDK>

Una vez tenemos esto, ya tendremos creado nuestro proyecto. Ahora sólo quedaría acceder a Eclipse e importar este proyecto, donde ya podremos compilarlo, debugearlo y más opciones de diseño.

En dicho proyecto se han cambiado algunos parámetros en el código para que la aplicación nos devolviese ciertos datos en el *log*. Estos datos nos permiten hacer un estudio de las variables que se requerirán para el estudio de la red.

Una vez creada la *.apk* correspondiente, se instalará en el terminal que se dispone y ya se podrán realizar las pruebas oportunas activando el modo debug.

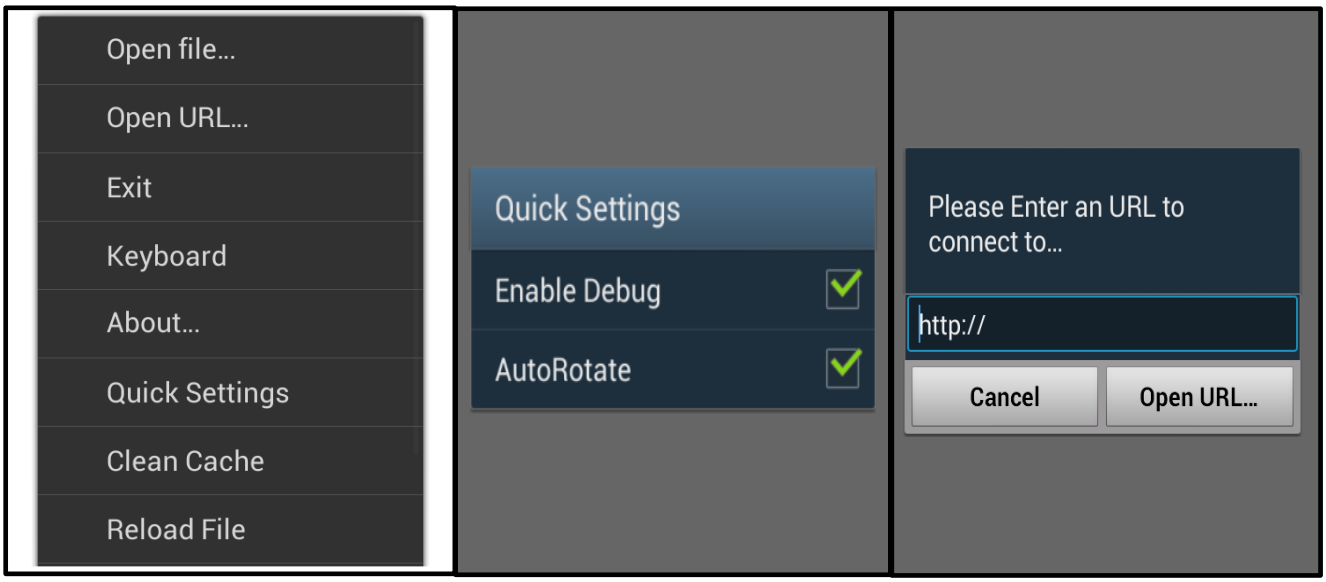


Figura 24 – Osmo4 en dispositivo Android.

II.2 Servidor Web.

Nuestro servidor web no es mucho más diferente que cualquier otro servidor apache. La característica que posee éste es que posee los archivos *.m4s* (particiones del vídeo original generadas por MP4BOX [2.II.1]) y los *.mpd* correspondientes que servirán al usuario para poder tener una amplia información.

Un servidor apache no es más que un servidor web HTTP de código abierto. Este tipo de servidores presenta, entre otras características, que son altamente configurables, bases de datos de autenticación y negociación de contenido.

Para el desarrollo de este proyecto no es necesario que el servidor tenga ninguna funcionalidad extra, ya que sólo necesita procesar peticiones HTTP que responderá según lo demandado (ficheros MPD y los segmentos de vídeo).

II.3 Proxy CCNx.

El proxy tiene la principal función de recibir peticiones HTTP GET y transformarlas a paquetes *Interest* [2.III.3] para que en toda la red sean capaces de verificar si ese segmento lo contienen en su caché o no. Como todo proxy nos proporciona caché, control de acceso, registro del tráfico, prohibición de cierto tipo de tráfico, etc.

En el caso de este proyecto se ha realizado toda la estructuración de la red CCNx virtualmente, no físicamente, reduciendo así costos. Como ya hemos visto, en el proxy CCNx se realizarían todas las peticiones con el formato ya comentado en el que el proxy elimina la primera parte del nombre (su propio nombre) y genera una petición con el resto. Gracias a esto se podrá acceder a la información de toda la red CCN y conseguir el segmento de vídeo deseado si se encuentra en la red. La versión de cada segmento se actualizará si aparece otro de las mismas características.

II.4 NetFetch.

NetFetch tiene un cometido muy parecido al que tiene el proxy CCNx, como ya hemos visto [3.I.2] reconvierte los paquetes *Interest* en peticiones HTTP. La gran diferencia que tienen es, principalmente, que el NetFetch siempre está en el lado del servidor y es el encargado de que no se eliminen los archivos guardados en la red. Para el reseteo de las cachés habrá que acceder al NetFetch y limpiarlas.

Más adelante veremos cómo se arrancan las redes CCN y como se pueden borrar las cachés en esta red construida virtualmente.

Parte 4
Validación

En esta parte de la tesis se hablará de las diferentes baterías de pruebas realizadas y los resultados de las mismas. Con esta información se podrá deducir los beneficios y desventajas de la arquitectura diseñada y si la solución que se ha proporcionado en este Trabajo Fin de Grado ha solventado las carencias que se planteaban al comienzo del mismo.

I Pruebas y resultados

Para el estudio de esta arquitectura hemos realizado una serie de batería de pruebas que a continuación se irán mostrando. También se va a explicar un poco como se ha ido desarrollando el proyecto y los inconvenientes encontrados en el mismo.

En primer lugar, como ya se comentó, se decidió realizar el proyecto con el reproductor de vídeo VLC, pero se decantó por realizarlo usando Osmo4, ya que nos proporcionaba una mayor posibilidad de configuración y adaptación de ciertos parámetros a nuestras necesidades ya que es libre y más ligero.

Por otro lado, tuvimos ciertos problemas con la versión de Android de la que disponíamos en el terminal HTC Legend, por lo que se procedió a rootearlo e instalar una ROM más actualizada que nos proporcionase el correcto funcionamiento de la última versión de Osmo4.

Puesto que disponíamos también de un emulador en Eclipse, se decidió hacer la batería de pruebas a través de esta herramienta que se nos ofrecía ya que no había constancia segura de que el terminal funcionase correctamente con la nueva versión de Android instalada (ya que el terminal era antiguo para la versión instalada de Android). Como se verá en los siguientes puntos, las pruebas con el emulador de Android no dieron los resultados esperados (primero se realizaron todas las pruebas con el Emulador y posteriormente con el terminal), ya que la aplicación corría muy lenta y tardaba en demandar los segmentos de vídeo. Por ello, se decidió realizar las pruebas en el terminal y ver si los resultados eran los esperados. En el último punto de esta sección se dirán las conclusiones obtenidas.

El vídeo de prueba consta de 11 segmentos. Cada segmento de vídeo que se ha utilizado para las pruebas tiene una duración de 5 segundos exceptuando el último segmento que tiene una duración de 1 segundo, ya que el vídeo streaming a reproducir tiene una duración de 51 segundos. Algunas de las pruebas con el terminal HTC se han realizado en movimiento para poder estudiar que sucede cuando conexión es baja o limitada o cuando se produce una desconexión momentánea.

Hay que destacar también que la red CCN no ha estado siendo utilizada por ninguna otra aplicación durante las pruebas. Esto es importante porque, si hay otros usuarios descargando otro contenido, dicho contenido competirá por la caché de la red CCN, que estará limitada a un cierto tamaño. Si fuese ese el caso, un contenido podría eliminarse de la cache, ya que al estar llena, se eliminan los contenidos que menos se han utilizado. En nuestro estudio no quisimos introducir esta variable, ya que los resultados no serían siempre los mismos y dependerían del instante de las sucesivas descargas.

En todos los casos, las pruebas se han realizado más de una vez para comprobar que los resultados no eran resultados sin constancia. En esta tesis únicamente exponemos una de las varias validaciones realizadas. De este modo, el lector podrá visualizar el resultado de las pruebas sin sobrecargar excesivamente la información del documento.

I.1 Pruebas sin red CCN.

Esta batería de pruebas se ha centrado en una red sin CCN, con un encaminador que procesa las peticiones HTTP GET del cliente y las envía al servidor.

Como en cualquier reproducción de un vídeo streaming DASH *[Figura 25]*, el cliente solicitará el fichero MPD para poder pedir el archivo de vídeo con las condiciones expuestas en este fichero de metadatos (1). El servidor web contestará al cliente devolviéndole dicho MPD con el que podrá comenzar la demanda del streaming multimedia (2). Para ello solicitará al encaminador correspondiente el segmento de vídeo deseado, este se lo comunicará al servidor (3) y este lo devolverá de vuelta (4).

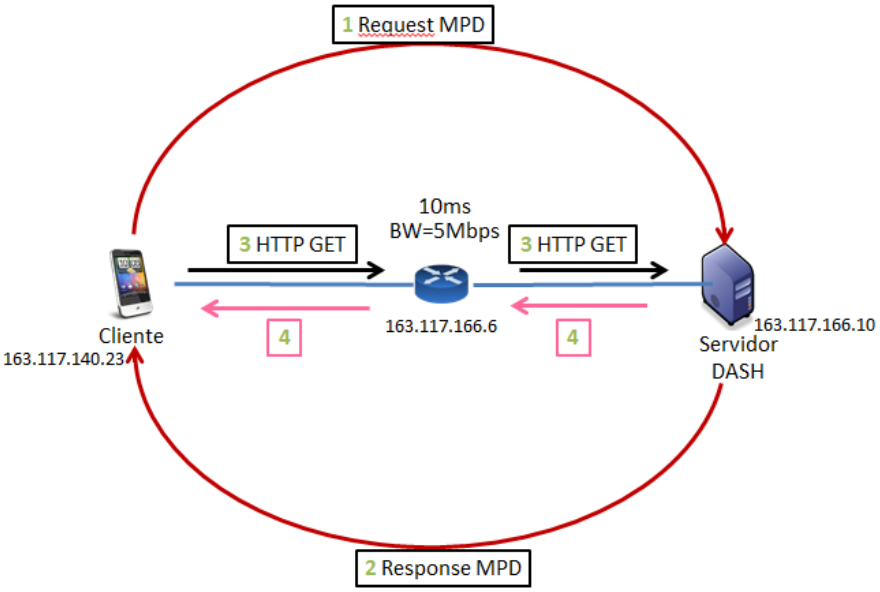
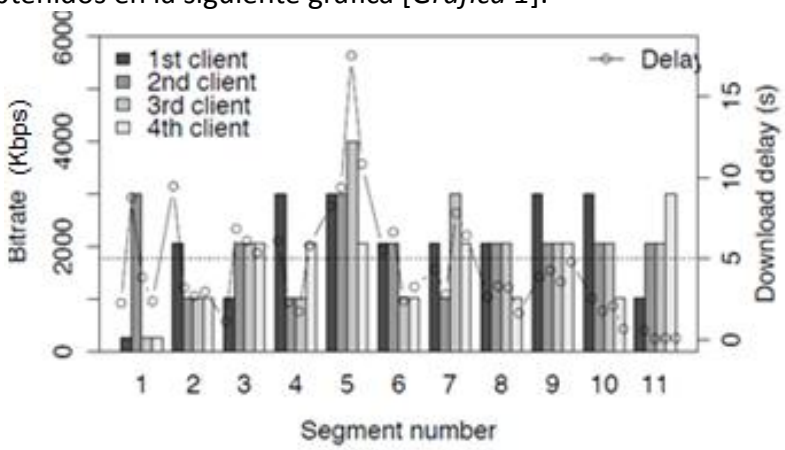


Figura 25 – Prueba sin red CCN.

Para la construcción de esta batería de pruebas se ha diseñado de tal manera que el ancho de banda sea de 5Mbps y que el encaminador reenvíe los segmentos con un delay (retardo) de 10ms a la vuelta del mensaje, para que los tiempos del estudio sean relevantes y puedan ser más intuitivos.

Emulador de Android.

Las primeras pruebas que se realizaron en este entorno fue mediante el Emulador de Eclipse del que ya se ha hablado [1.IV.2]. Estas pruebas condujeron a pensar en que el Emulador de Android no realizaba correctamente el pedido de paquetes como posteriormente se comprobó. Vamos a observar detenidamente los resultados obtenidos en la siguiente gráfica [*Gráfica 1*]:



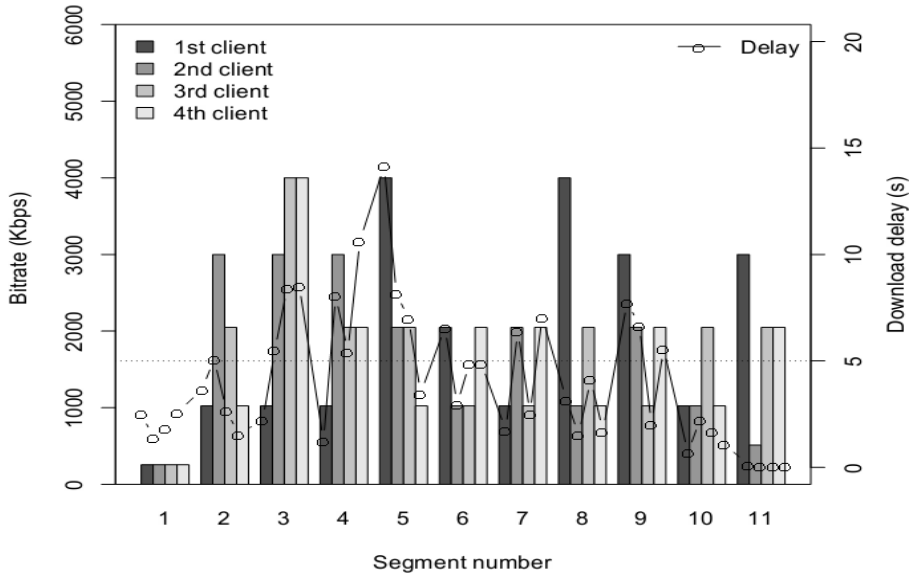
Gráfica 1 – Prueba sin red CCN, Emulador Android.

En primer lugar se va a comentar un poco la estructura de esta gráfica y qué representa, para que en las próximas sea más sencilla su comprensión. Estas gráficas representan la petición de 4 clientes que están conectados desde el punto Wi-Fi, y los cuales demandan un cierto vídeo concreto en diferentes momentos de tiempo. En el 'eje x' se representa cada uno de los 11 segmentos de los que está compuesto el vídeo, mientras que en el 'eje y' de la izquierda, y representado por barras, se tiene la calidad descargada mientras que en el 'eje y' de la derecha tenemos el retardo obtenido para descargar el segmento correspondiente. Las calidades con las que se han trabajado son de *256Kbps, 512Kbps, 1024Kbps, 2048Kbps, 3000Kbps y 4000Kbps*. El objetivo es por tanto, ver el retardo que se obtiene en cada demanda y la calidad de la petición.

Como se puede apreciar, para este tipo de pruebas el cliente va demandando una calidad u otra en cada segmento según los tiempos que hayan tardado los anteriores en llegar. Al tener un ancho de banda limitado los segmentos de una calidad superior a 3000Kbps tienen un delay considerable, teniendo en cuenta que cada segmento es de 5 segundos. Por lo que se concluye que en esta arquitectura de red y con el ancho de banda asignado, los segmentos con una calidad inferior a 3000Kbps serán reproducidos por el cliente sin que se detenga en ningún momento.

HTC Legend.

Después de haber completado TODAS las pruebas con el Emulador de Android (no sólo ésta, sino también con la red CCN implantada) se procedió a realizarlas con el HTC Legend. Para una mejor estructuración de esta tesis, se va a proceder a comentar los resultados obtenidos con el HTC Legend para poder compararlas a su vez con las del Emulador.

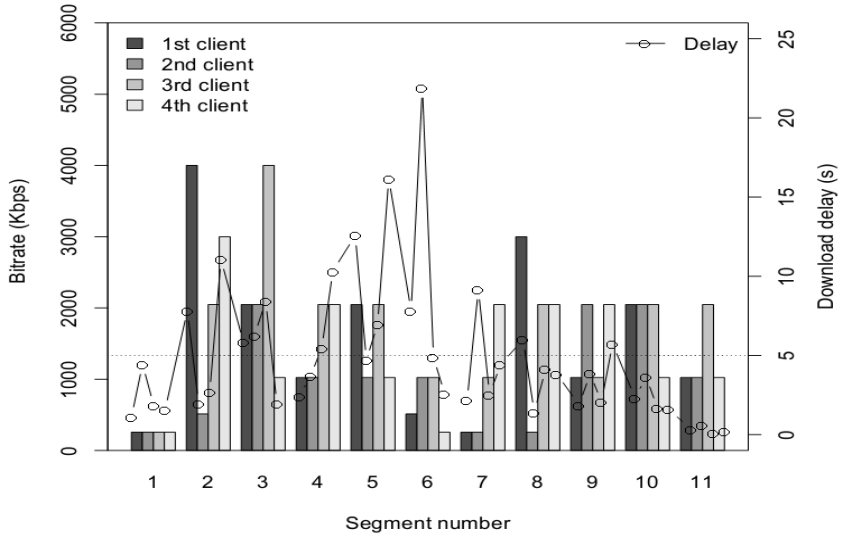


Gráfica 2 – Prueba sin red CCN, HTC Legend.

Como se aprecia, la diferencia entre lo obtenido entre el emulador y el terminal no es muy grande, puesto que no es una red muy compleja y no da muchos saltos para llegar al servidor y además de que siempre es la misma ruta (siempre se accede al servidor para cada segmento).

HTC Legend en movimiento.

Otra de las pruebas realizadas consiste en probar cómo se comporta tanto el terminal como la red ante pérdidas en la conexión a la red o que dicha cobertura sea de un nivel bastante pobre. Se puede observar que la única diferencia reside cuando se pierda la conexión que tardaremos más en reproducir ese segmento (hasta que se reconecte y consiga descargarlo).



Gráfica 3 – Prueba sin red CCN, HTC Legend en mov.

I.2 Pruebas con red CCN.

En este tipo de pruebas hemos optado por limitar el ancho de banda y poner el enrutador en el lado del cliente para que nos aporte el delay de 10ms. El primer paso de todos será como en todo proceso de petición de un streaming multimedia basado en el estándar DASH, la petición del MPD (1), con su respectiva respuesta y envío del mismo. Una vez el cliente disponga de la información, como ya sabemos, se pedirá al servidor la descarga de los segmentos correspondientes (3), se los proporcionará (4) y quedarán almacenados en la red CCN. Una vez otro usuario quiera ese mismo segmento (5) se los devolverá aquel que los posea. La arquitectura es la siguiente:

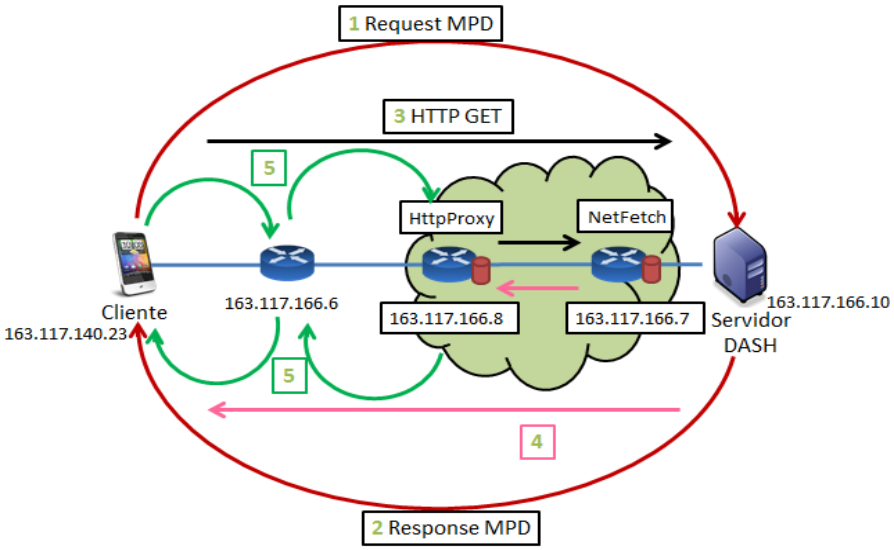
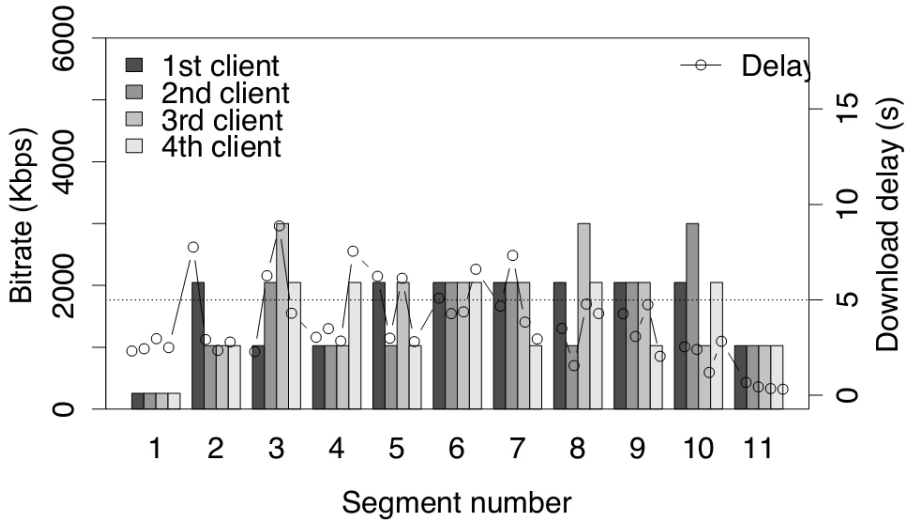


Figura 26 – Prueba con red CCN.

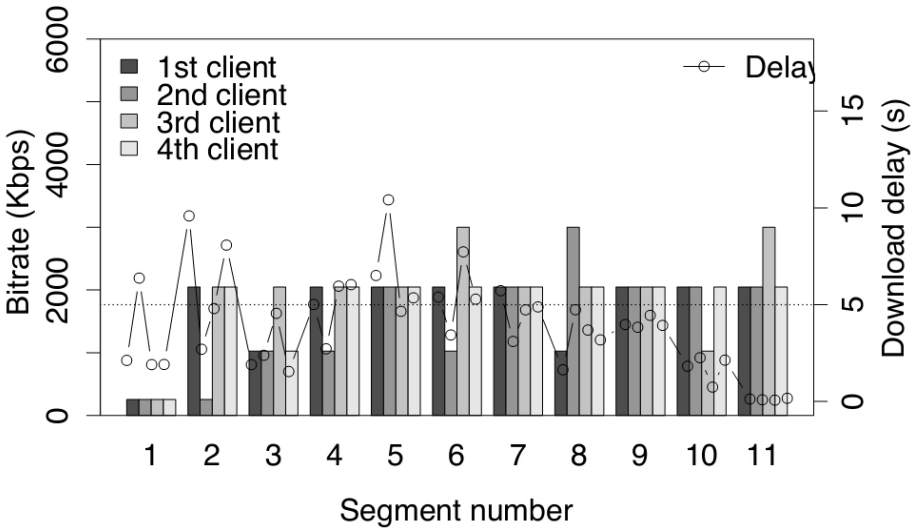
El ancho de banda para esta prueba corresponde con el mismo que la anterior, ya que utilizaremos el mismo encaminador con las mismas características, 5Mbps de ancho de banda y 10ms de delay.

Emulador de Android.

Esta batería de pruebas se realizaron 2 veces en el caso del Emulador porque como bien se ha comentado, se tenían ciertas dudas sobre el funcionamiento correcto del mismo.



Gráfica 4 – Prueba 1 con red CCN, Emulador Android.



Gráfica 5 – Prueba 2 con red CCN, Emulador Android.

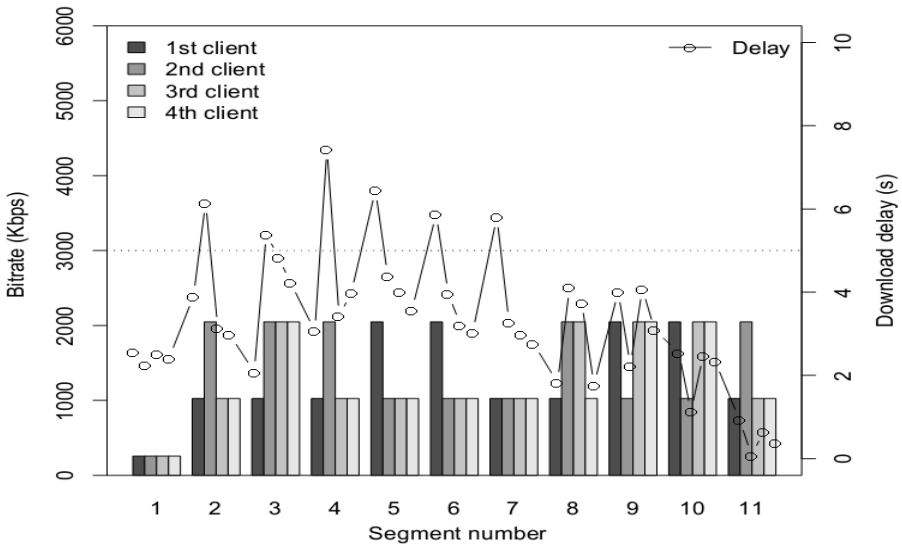
Se ve que en muchos casos, aunque se tenga el segmento almacenado en la caché de las redes CCN, aumentan su tiempo de descarga. Por ejemplo, donde más claramente se ve en la Prueba 1 [*Gráfica 3*] es en el segmento 1. El primer cliente se descarga este segmento con una calidad de 256Kbps, el segundo cliente decide también descargarse el primer segmento con dicha calidad. Si la red o el Emulador funcionasen correctamente, el tiempo disminuiría, puesto que ya no sería necesario

acceder al servidor para obtenerlo. En el segmento 5 de la misma prueba también nos percatamos de lo mismo. En la segunda prueba también se puede observar este hecho. Aunque con algunos segmentos sí que se devolvían en menos tiempo cuando ya se tenía almacenado en la red, véase el caso de la prueba 2, segmento 2 por ejemplo.

* Para estudiar estos errores, se procedió a acceder al NetFetch para comprobar si era debido a que la red CCN no almacenaba correctamente los segmentos y se accedía hasta el servidor para proveerlos o era por el Emulador. Rápidamente nos percatamos que no accedía al servidor, sino que era el Emulador que realizaba las peticiones más tarde de lo que debía.

HTC Legend.

Como se verá en la siguiente gráfica, en este tipo de pruebas se ve claramente como los segmentos almacenados tardan menos en llegar al cliente que los solicita. Esto nos muestra que el cometido de la red virtual montada realiza como se esperaba su cometido.



Gráfica 6 – Prueba con red CCN, HTC Legend.

Como se aprecia, todos los segmentos almacenados en la red por clientes anteriores son servidos en un menor tiempo a dicho cliente. Esto se debe a que al no pasar por el servidor nos evitamos dar más saltos para acceder al contenido multimedia.

I.3 Segundas pruebas con red CCN.

Para seguir con la batería de pruebas y corroborar el buen o mal funcionamiento de contenido multimedia con formato DASH sobre redes CCN, se procedió a realizar la siguiente prueba. En este testbed se elimina el encaminador y se procede a cambiar el ancho de banda en el lado del servidor por 1, 2 y 5Mbps para comprobar como dependiendo del ancho de banda se demanda más o menos calidad en el vídeo streaming. La arquitectura implementada es la siguiente:

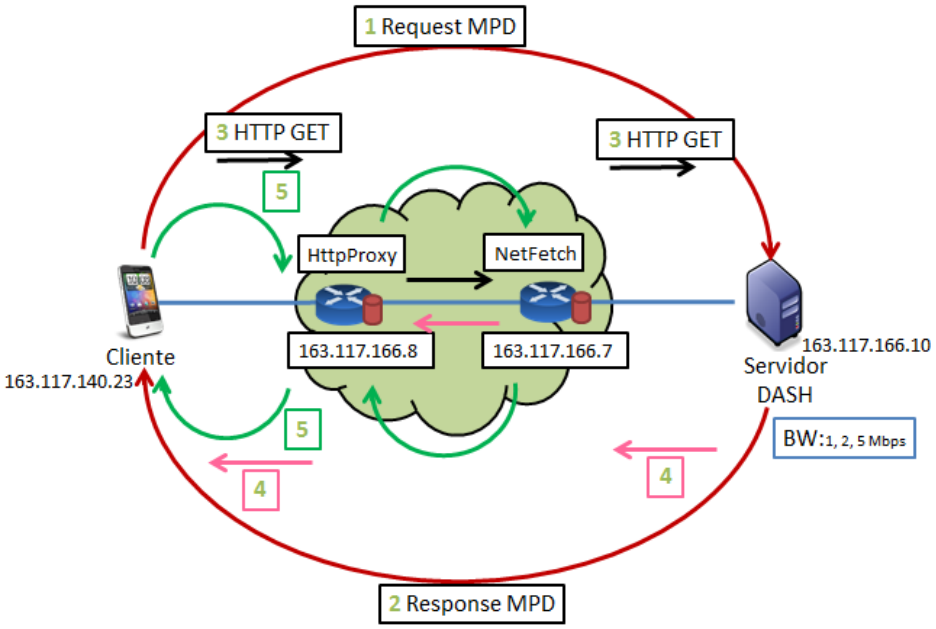


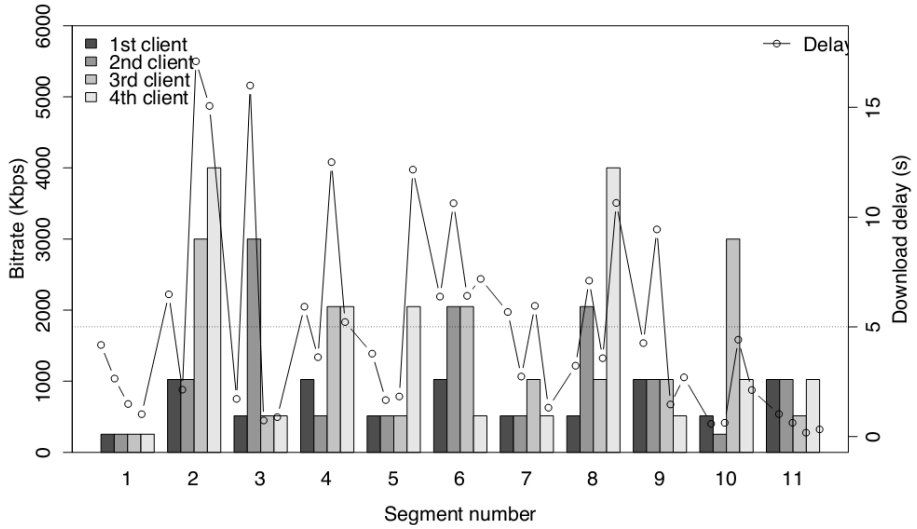
Figura 27 – Prueba con red CCN.

- **BW: 1Mbps:**

Emulador de Android.

En este testbed se comprueba como para calidades demasiado altas la descarga de esos segmentos son demasiado elevadas, incluso doblando la duración del segmento. Por lo que en la siguiente petición el cliente decidirá obtener un segmento de menor calidad para poder descargarlo y reproducirlo antes de que el anterior haya terminado su reproducción. Como ya se viene diciendo durante todo esta parte, estas pruebas con el Emulador nos siguen confirmando que no hace las peticiones cuando debe.

Es importante destacar que el algoritmo que utiliza nuestra aplicación de reproducción multimedia consiste básicamente en la elección de qué calidad de segmento descargarse. Cuando la aplicación observa que un segmento es proporcionado en un tiempo lo suficientemente reducido (por lo general, menor que el tiempo de duración de dicho segmento) hará una petición de otro segmento con una calidad mayor.

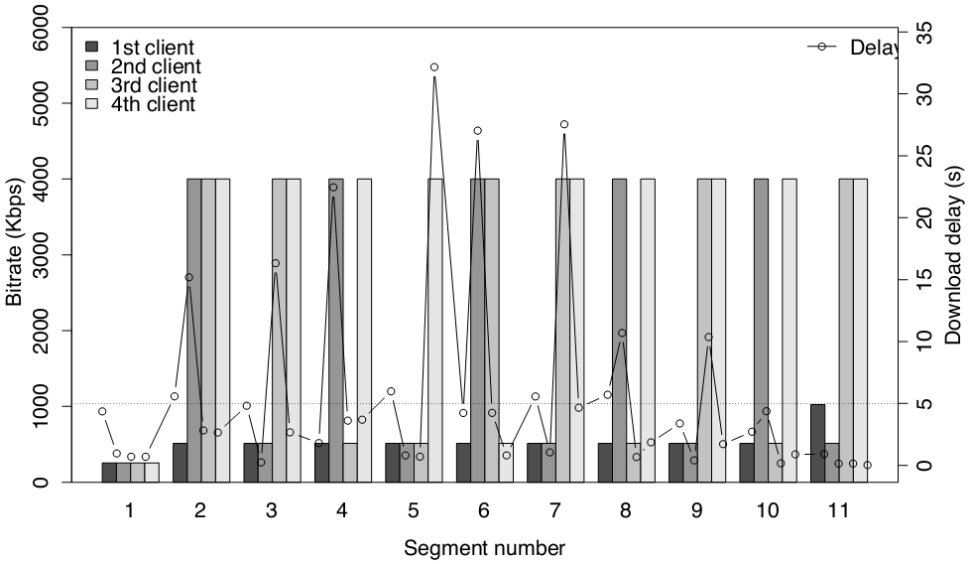


Gráfica 7 – Prueba exclusivamente con red CCN -1Mbps- Emulador Android.

HTC Legend.

En este testbed se aprecia muy claramente la gran ventaja de utilizar este tipo de redes. Esta prueba se va a proceder a explicar de un modo un poco más detallado. Al tener un ancho de banda tan limitado la calidad de los archivos multimedia no podrán ser muy elevados, ya que estos demandan bastante bitrate. El primer cliente que procede a demandar el vídeo comienza pidiéndolo con una calidad de 256Kbps (asignado por defecto pedir primero esta calidad por el algoritmo de nuestra aplicación). Al ver que el primer segmento tarda un poco menos en llegarle que la duración del mismo (5 segundos) decide demandar el siguiente segmento con una calidad mayor (no demasiada ya que el anterior tampoco fue mucho menor). Para este segundo segmento se exige una calidad de 512Kbps, al no tardar mucho más que la duración del vídeo, la aplicación prueba con el siguiente segmento con la misma calidad. Resumiendo, el primer cliente al no tener ningún segmento almacenado en la red, siempre deberá llegar hasta el servidor para que sea recibido. Como hay un ancho de banda limitado en el lado del servidor, la calidad nunca podrá ser superior a 1.024Kbps, puesto que el ancho de banda es de 1Mbps.

Una vez el segundo, tercero o cuarto cliente desean realizar la petición del vídeo streaming que ya se solicitó ocurre lo esperado. En el primer segmento, como en el algoritmo está predefinido que sea de la calidad más baja y el anterior cliente también hizo esa misma petición, no aseguramos que se encontrará en la red almacenado. Por lo que para este primer segmento el tiempo de retardo será mínimo. Debido a esto, los posteriores clientes siempre demandarán un segmento de una calidad bastante superior a la mínimo (por lo general, al tener tan poco delay, la mayor calidad disponible, 4.000Kbps). Para el primer cliente que demande esta calidad (el segundo cliente) verá como su tiempo de retardo aumenta considerablemente ya que no está almacenado y tendrá que ir hasta el servidor, el cual está limitado en ancho de banda, por lo que reducirá considerablemente su calidad en el próximo segmento.



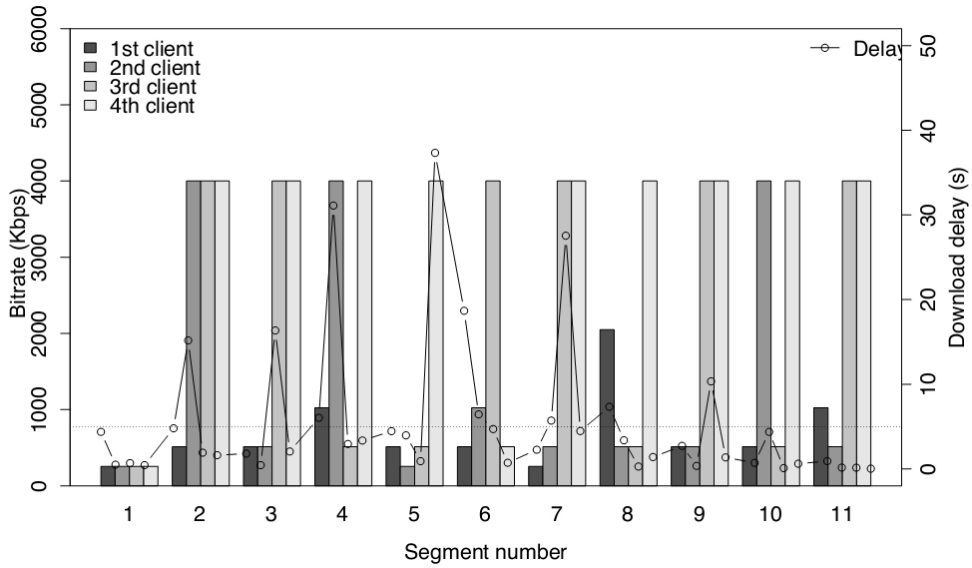
Gráfica 8 – Prueba exclusivamente con red CCN -1Mbps- HTC Legend.

El gran beneficio que nos aporta esta arquitectura diseñada viene siendo que, cuando en el lado del servidor hay un límite de ancho de banda bajo, una vez se vaya demandando el vídeo varias veces, habrá un punto en el cual ya no será necesario hacer peticiones al servidor y no nos influirá esa limitación en el ancho de banda.

Este ejemplo se podría escalar a arquitecturas más complejas, donde para acceder desde la red CCN al servidor existiesen varios intermediarios que nos pudiesen provocar un retraso en la llegada de nuestros segmentos o incluso que en algunos de estos intermediarios hubiesen anchos de bandas bajos, como ocurre en este ejemplo.

HTC Legend en movimiento.

En esta batería de pruebas se ha decidido provocar caídas de conexión en la red inalámbrica del dispositivo móvil y niveles bajos de cobertura.



Gráfica 9 – Prueba exclusivamente con red CCN -1Mbps- HTC Legend en movimiento.

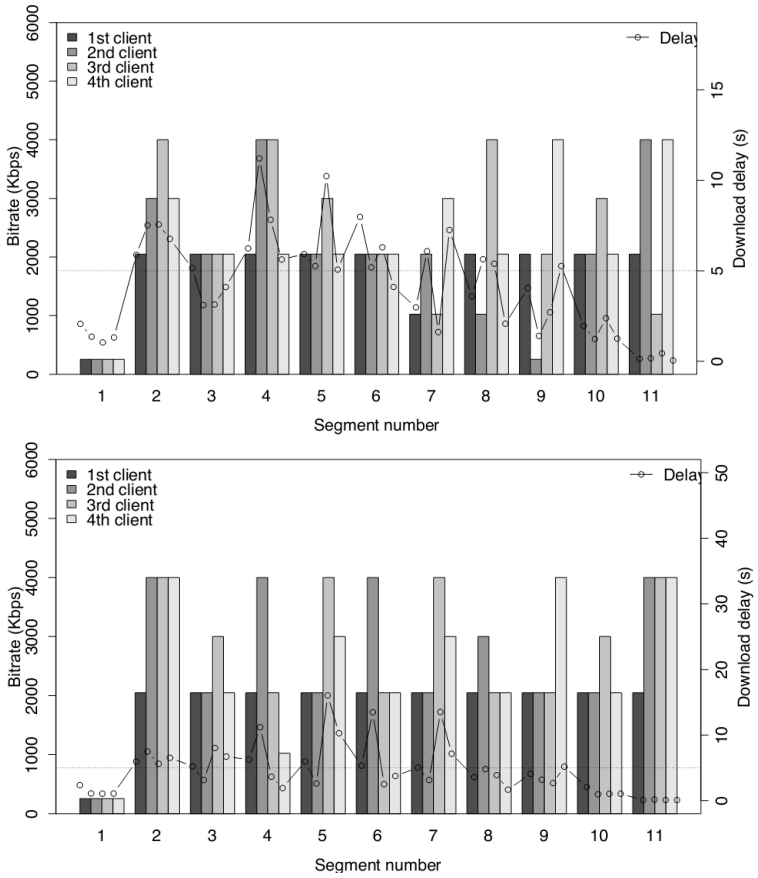
En estos testbed se aprecia como cuando el cliente tiene una caída de señal inalámbrica y se desconecta de la red, la aplicación no reconoce la perdida de conexión. Cuando nos reconectamos de nuevo, la aplicación observa que se ha incrementado mucho la recepción del segmento, debido a la desconexión, y en la siguiente petición se disminuirá la calidad considerablemente. Probablemente si la aplicación reconociese nuestra desconexión y únicamente calculase el tiempo que tarda el segmento en llegar, en la petición del siguiente segmento no disminuiría la calidad y tuviese que acceder hasta el servidor. Esto provocaría que en la red se volviesen a almacenar los segmentos de calidad más baja.

- **BW: 2Mbps:**

Emulador de Android.

A partir de este punto, sólo se pondrán las gráficas obtenidas con el Emulador en las diferentes calidades, ya que como se ha dicho, no aportan nada al estudio, puesto que su funcionamiento no es correcto. Por lo tanto se dejan las correspondientes gráficas para que quede constancia del estudio realizado y que en futuros trabajos no se realicen con dicho emulador.

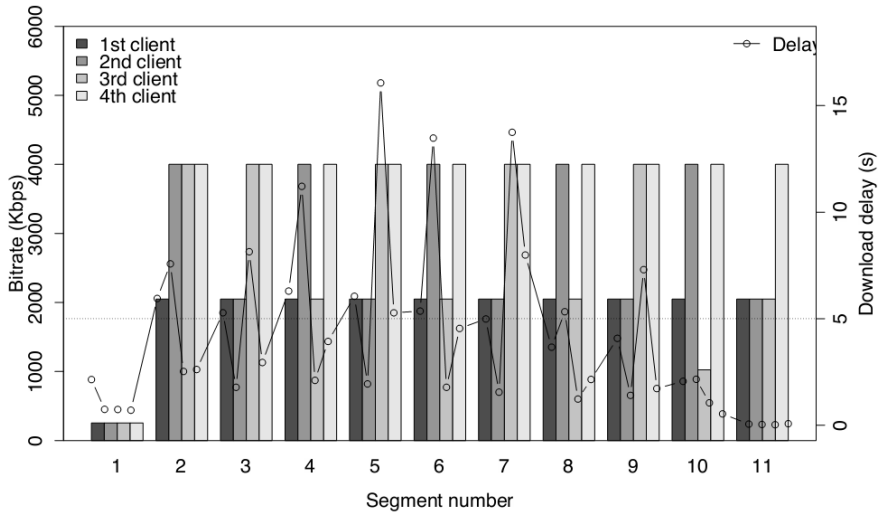
Para las pruebas con 2 y 5Mbps se realizaron dos veces los mismos testbed para hacer un estudio más a fondo del problema. Como se puede analizar, no existe ninguna concordancia con ninguna de ellas, mientras que con las del HTC se realizaron más de una y si existían.



Gráfica 10 – Pruebas exclusivamente con red CCN -2Mbps- Emulador.

HTC Legend.

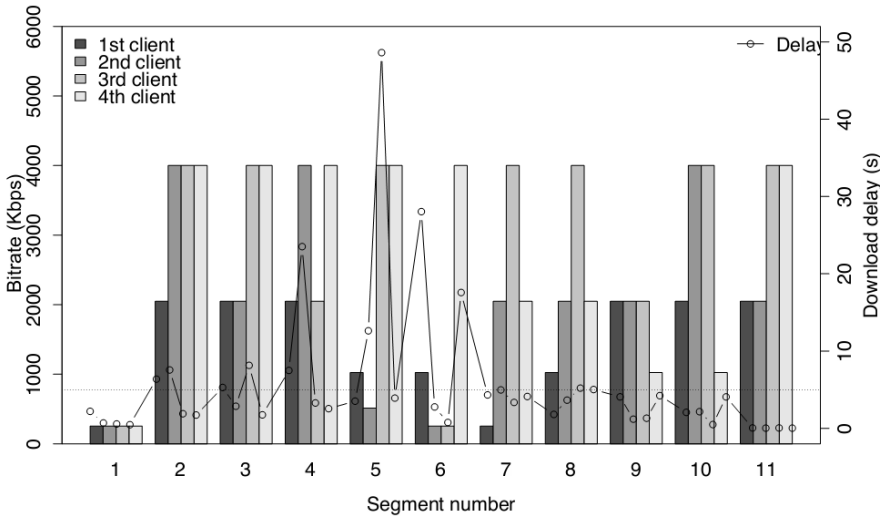
En esta batería de pruebas se observa que, como en la anterior prueba[*] el principal afectado en este tipo de testbed es el primer cliente. Debido a que tiene que ir siempre hasta el servidor para conseguir los segmentos necesarios para la reproducción del vídeo. Como tenemos un ancho de banda de 2Mbps, afectará a la calidad del cliente siempre que tenga que obtener los segmentos del servidor. Con este ancho de banda, aquel que solicite un vídeo del servidor, por lo general, obtendrá un segmento con una calidad inferior a 2.048Kbps. El resto de las peticiones de otros clientes es exactamente igual que en la ocasión anterior. Siempre que se obtenga un segmento en un tiempo inferior a la duración del mismo se hará una petición del siguiente con una calidad superior. En este caso se deduce, también, que realizadas una cantidad suficiente de peticiones, en la red se almacenará el vídeo con la calidad máxima con tiempos de espera ínfimos.



Gráfica 11 – Prueba exclusivamente con red CCN -2Mbps- HTC.

HTC Legend en movimiento.

En este caso vuelve a ocurrir lo mismo que en el anterior[*].

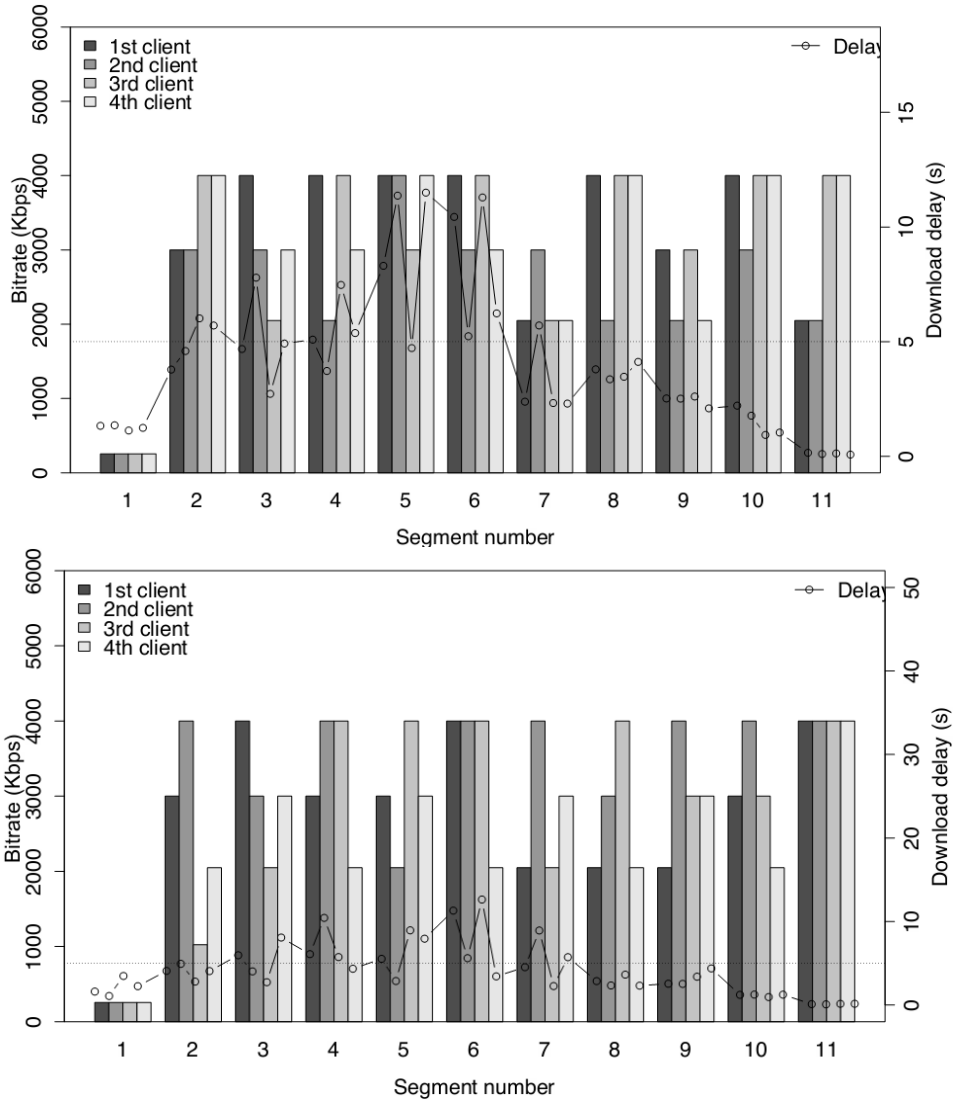


Gráfica 12 – Prueba exclusivamente con red CCN -2Mbps- HTC en movimiento.

Cuando se produce una caída (sin que expire la conexión con la red), reducirá al mínimo la calidad para poder paliar con el retraso tan elevado que se sufrió con ese segmento. Una vez llega la siguiente trama, con procedencia del servidor, la aplicación se dará cuenta de que se tardó menos de lo esperado y volverá a incrementar la calidad hasta el máximo que permita el ancho de banda del lado del servidor, puesto que a partir de la caída siempre tendremos que acudir al servidor.

- **BW: 5Mbps:**

Emulador de Android.

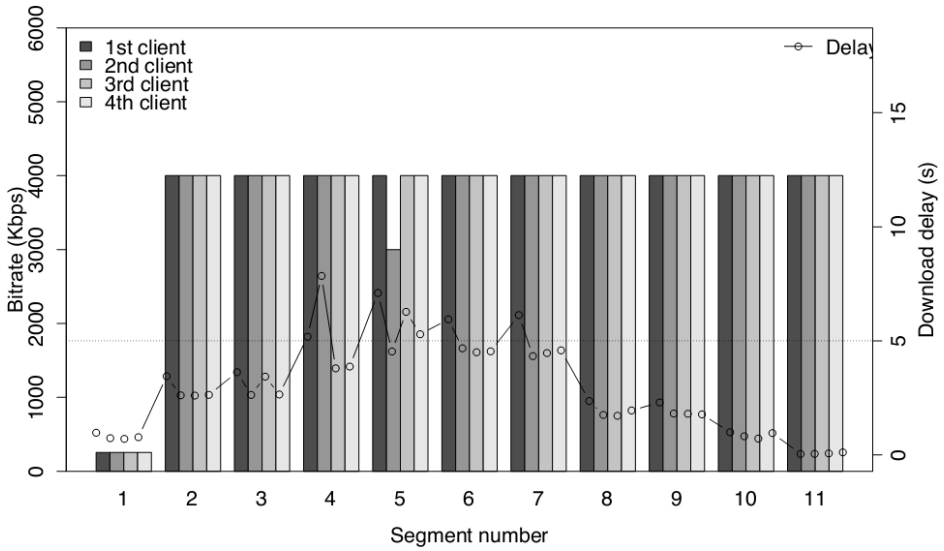


Gráfica 13 – Pruebas exclusivamente con red CCN -5Mbps- Emulador.

HTC Legend.

En esta batería de pruebas el ancho de banda es suficiente como para poder transmitir cualquier tipo de calidad, hasta la más alta (4.000Kbps), por nuestro canal. Debido a esto, la única diferencia que se puede observar si lo comparamos con una red similar pero sin CCN, son los tiempos de retardo. Puesto que con las redes CCN no nos

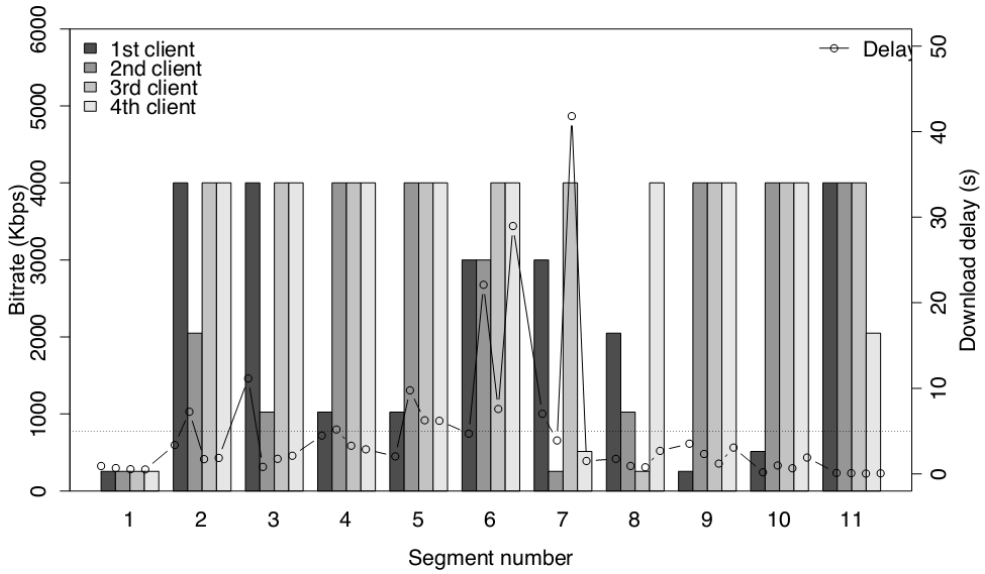
hará falta ir hasta el servidor una vez tenga almacenados los segmentos, nos evitaremos saltos y recorrido. Se concluye por lo tanto, que con las redes CCN, se reduce el delay en conexiones suficientemente altas para el soporte de la calidad máxima.



Gráfica 14 – Prueba exclusivamente con red CCN -5Mbps- HTC.

HTC Legend en movimiento.

En este caso el único problema que se presenta es cuando se cae la conexión que reducirá la calidad en el siguiente segmento debido a la espera tan elevada. Una vez sirva el segmento con una calidad reducida, como se posee de un ancho de banda que no limita la calidad, el siguiente segmento tendrá la máxima calidad.



Gráfica 15 – Prueba exclusivamente con red CCN -5Mbps- HTC en movimiento.

Se concluye, así, que sigue beneficiando este tipo de arquitectura puesto que se evitan saltos cuando no existen caídas.

Parte 5
Conclusiones y trabajos futuros

En esta parte de la tesis se hará un balance del estudio realizado. Se analizará, partiendo de los resultados obtenidos, si el trabajo realizado mejora otras estructuras y redes similares como P2P y CDN con sus debidas conclusiones. Por último, se presentarán los posibles trabajos futuros para aumentar la eficiencia tanto de la red como de la aplicación utilizada.

I Conclusiones

En este apartado se presentan las conclusiones sacadas basadas en las investigaciones, experimentos y pruebas realizadas. Se concluye por tanto varios puntos importantes:

- ✓ DASH se puede adaptar con cierta facilidad a un entorno CCN y aprovecharse de las características de almacenamiento. Fácil adaptación en parte gracias, a la propiedad de partición de la que dispone DASH, que facilita el almacenamiento en caché de archivos no muy pesados.

- ✓ Se demuestra también que DASH sobre redes CCN (DASC), sigue siendo compatible con las infraestructuras HTTP gracias a los HttpProxy y los NetFetch que las redes CCN nos proporcionan.

- ✓ El rendimiento de DASC converge progresivamente a la mejor calidad posible, gracias a, que como se ha visto en los experimentos, debido a que se realizan menos saltos con la red CCN la aplicación demanda calidades cada vez superiores.

- ✓ Se producen mayores beneficios con DASC cuando el ancho de banda limitado se encuentra en el lado del servidor, puesto que con las redes CCN si ya se tiene almacenado ese segmento nunca alcanzaremos el servidor. Mientras que cuando el ancho de banda limitado se encuentra en el lado del cliente los beneficios son menores ya que el tiempo ganado es el que se pierde en llegar hasta el servidor y la calidad del vídeo se verá influenciada por este bitrate.

- ✓ Reduce la congestión y mejora la velocidad de entrega de los segmentos.

- × Cuando se producen caídas de red en el dispositivo y exista un ancho de banda limitado, la calidad se verá influida ya que los posteriores segmentos tendrán que ser entregados por el servidor. Esto se debe a que la aplicación pensará que la red está congestionada o limitada con un ancho de banda menor y que los demás segmentos también sufrirán estos delays.

II Líneas de trabajo futuras

El presente proyecto ha servido para conocer y analizar el estándar DASH y como volcarlo a las redes CCN. Para ello ha sido necesario un dispositivo móvil con Android para la instalación de la aplicación utilizada y una red virtual que simulase el funcionamiento de CCN. Teniendo esto en cuenta se muestran las principales mejoras que se podrían aportar en un futuro.

En cuanto a la aplicación utilizada, **Osmo4**, están son las principales mejoras que se han pensado:

- ↳ Mejorar la interfaz gráfica. Osmo4 tiene una interfaz gráfica muy reducida y no permite realizar cosas tan básicas como parar un vídeo streaming en reproducción.

- Información deseada. Ya que es una herramienta de desarrollo, una mejora sería que proporcionase al usuario qué tipos de datos desea recoger y dónde les gustaría almacenarlos.

- Multiplataforma. Probar dicha aplicación en otras plataforma móviles no sólo Android.

En cuanto a la estándar DASH y las redes CCN, se han pensado en los siguientes trabajos futuros:

- Ampliar la infraestructura de la red. Este aspecto se podría realizar de una manera muy simple añadiendo más nodos a la red para realizar un estudio de cómo los nodos se comunican entre sí y poder analizar como realizan entre ellos las peticiones de vídeos almacenados.

- Mayor inteligencia al sistema. En este punto nos referimos a que a la hora de proceder en nuestras pruebas para que la red CCN reconociese al cliente teníamos que añadirlo nosotros directamente y en el caso de DASH a la hora de crear los ficheros de metadatos (MPD) añadíamos la ruta que debían de seguir.

- Manejar la funcionalidad de un servicio más avanzado. Como la base de datos de aplicaciones pesadas y los medios de comunicación social, no sólo archivos multimedia. Para esta mejora se necesitará conocer la información de estado y poder ser trasladado junto con el servicio.

- Realizar pruebas con un dispositivo móvil conectado a una red 3G o 4G para estudiar el comportamiento en movimiento sin pérdida de la señal

- Implementar un nueva red que permita al HttpProxy decidir si pasar o no por la red CCN dependiendo si está en sus registros o no. Si no está en sus registros iría al servidor sin saltar por toda la red CCN. Véase la *Figura 28*.

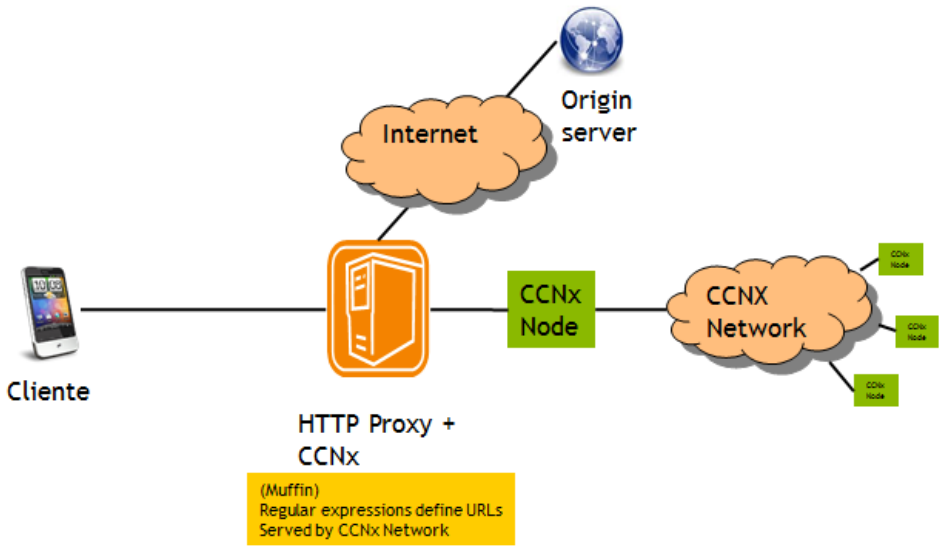


Figura 28 – Red alternativa con CCNx.

I Incentives

The main objective of this thesis is to design a content delivery network to distribute adaptive video streaming, which will be based on the DASH protocol. This design will be validated throughout a set of trials on top of a dedicated testbed, in order to analyze the performance that can be achieved with this architecture over a wireless network. Consequently, we will discuss about the consumption of videos in our society. Because of this, we have studied the behavior of them and how to improve our playing time. As we know, this kind of videos is characterized by the customer has a data buffer that stores the downloaded material for later display. Therefore, we have investigated how to reduce the sending of video frames so that the user has these video frames as soon as possible. If the customer has a fragment and the next doesn't have it, the video will stop until the user can obtain the following fragment. Thanks to our research, we have analyzed and studied how they behave CCNx networks with streaming videos (based on the DASH standard). In the following chapters, we will show the result of this analysis (if the video loads in less time) adapting these tests to an Android device that it can play videos according to the DASH standard.

IV Purposes

In this section, we will describe the aims we want to achieve in this thesis. First, we will talk about the first objectives and subsequently, we will discuss the technical objectives that we want to accomplish.

IV.1 Initial Purposes

The main objective of this thesis is to analyze the functioning of CCNx networks using the DASH standard videos. With these networks, we play the videos in less time. When the server has the first request of this video, these networks can be stored fragments, the next time, you will be able to play the video, the server already will not send anything. The CCNx network will have these fragments and send us without more jumps by the network. As we shall see later, the video file will have a specific format that will contain all the information necessary to contact the server. It can choose the best quality and other information included in this format.

Once studied these first goals, we have studied how to structure virtually CCNx network and the possibility to play any type of video in DASH format with an Android device.

IV.2 Technical Purposes

In this section, we will talk about more technical and specific objectives that we were considering in the project. We divide them into several points:

- **Meet CCNx networks.** The first objective was to learn about the operation of these networks. We are looking for information on the official website and

other materials to understand the operation. Later, we were looking for the way to integrate it with the DASH standard. Then, we turn to the following points.

- **Master the environment.** As we have seen above, the study will be made with an Android device so we will have to meet and get acquainted with this programming language. An important part of this language is the Software Development Kit (SDK), which allows you to test, build, and debug applications, because this is a set of software development tools.
- **Emulator.** Once we have the application running, we have used the emulator of Eclipse (Android Emulator) and the use of the DDMS (Dalvik Debug Monitor Service). The DDMS is a tool for debugging in Eclipse, consists of five functional parts [*Figura 7*]: handling of tasks (1), where you will find emulators and phones that you have connected and their bodies; file management (2), interaction with the emulator (3), systems of Log (4) and (5) screenshots.

I Conclusions

In this section, we describe the results of the research, experiments and tests performed. Showing several important points:

✓ DASH can be adapted easily to a CCN environment and take advantage of the storage characteristics. Easy adjustment thanks to the partition of property available DASH, which provides caching of not very large files.

✓ We show that DASH over CCN (DASC) networks, supports HTTP infrastructure thanks to NetFetch HttpProxy and the CCN networks provide it.

✓ DASC performance gradually converges to the best possible quality, because as we have seen in the experiments, the less CCN jumps are made, the higher grades the application demands.

✓ DASC greatest benefits occur when limited bandwidth is on the server side, because the CCN network if you already have stored that segment never go to the server. While when limited bandwidth is on the client side the benefits are lower because the time gained is lost in reaching the server and the video quality will be influenced by this bitrate.

✓ Reduces congestion and improves the speed of segments delivery.

× When network downtime on the device and there is a limited bandwidth, the quality is affected because these segments have to be delivered by the server. This is because the application might think that the network is congested or limited with lower bandwidth and other segments also have these delays.

Parte 6
Glosario

ADT - Android Development Tools
CCN - Content Centric Networking
CDN - Content Delivery Network
DASC - DASH over CCN
DASH - Dynamic Adaptive Streaming over HTTP
DDMS - Dalvik Debug Monitor Service
GPAC - Proyecto multimedia de contenido multimedia
HTTP - Hypertext Transfer Protocol
IBC - Feria de contenidos multimedia
ISO/OSI - Modelo de interconexión de sistemas abiertos
ISOBMF - ISO base media file format
LASeR - Formato binario de presentación de codificación/decodificación
MP4Box - Empaquetador multimedia
MPD - Media Presentation Description

MPEG - Moving Picture Expert
MPEG-2 TS - MPEG-2 Transport Stream
MPEG-4 XMT - Idioma de presentación de codificación/decodificación
MPEG-4BIFS - Formato binario de presentación de codificación/decodificación
Osmo4 - Reproductor multimedia que soporta DASH
P2P - Peer-to-Peer
PARC - Palo Alto Research Center
RTCP - RTP Control Protocol
RTP - Real-time Transport Protocol
RTSP - Real Time Streaming Protocol
SDK - Software Development Kit
TCP - Protocolo de Control de Transmisión
TFG - Trabajo Fin de Grado
UDP - User Datagram Protocol
URI - Uniform Resource Identifier
URL - Localizador de Recursos Uniformes
W3C SVG - Idioma de presentación de codificación/decodificación
Wi-Fi - Wireless Fidelity
XML - eXtensible Markup Language

Parte 7
Bibliografía

[1] *CISCO.* Global Mobile Data Traffic Forecast Update, 2012–2017. [Internet]: http://www.cisco.com/web/ES/about/press/2013/2013-02-05-trafico-global-de-datos-moviles-se-multiplicara-por-trece-en-2017.html

[2] *I. C. Society*, "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", 2012.

[3] *CISCO*, "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2012–2017," 2013.

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

[5] *ITEC* – DASH over CCN. [Internet] http://www-itec.uni-klu.ac.at/dash/?p=1126

[6] *MPEG DASH Industry Forum* - Overview of MPEG-DASH Standard. [Internet]
http://dashif.org/

[7] Christian Timmerer and Christian Muller, Alpen –Adria Universitat Klagenfurt
Presentation on *"Dynamic Adaptive Streaming over HTTP (DASH)"*
Multimedia Systems – The Conference Series (MMSys '11)- 02, May 2011. [Internet]
http://www.slideshare.net/christian.timmerer/dynamic-adaptive-streaming-over-http-dash

[8] Information Technology – MPEG System Technology – Part 6: Dynamic Adaptive Streaming over HTTP (DASH).
ISO /IEC JTC 1/SC 29, Dated 2011-01-28
ISO /IEC FCD 23001-6

[9] Truong Cong Thang, Jin Young Lee, Jung Won Kang, Seong-Jun Bae, Sang Taick Park
Broadcasting & Telecom Convergence Research Laboratory, Electronics and Telecommunications Research Institute (ETRI), Korea
Paper on "Signaling Metadata for Adaptive HTTP Streaming"
Submitted to Multimedia Systems – The Conference Series (MMSys) ACM Multimedia Systems 2011 February 23-25, 2011 San Jose, California

[10] Thorsten Lohmar Markus & Kampmann - Ericsson GmbH; Per Fröjdh & Torbjörn Einarsson - Ericsson AB; Frédéric Gabin - Ericsson France
Paper on "Dynamic adaptive HTTP Streaming of Live Content"
Presented at 12th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2011 at the IMT Institute for Advanced Studies Lucca, central Italy, on June 20-23, 2011.

[11] Palo Alto Research Center, Inc. – Project CCNx. [Internet]
http://www.ccnx.org/

[12] Stefan Lederer, Christopher Müller, Benjamin Rainer, Christian Timmerer, and Hermann Hellwagner, "An Experimental Analysis of Dynamic Adaptive Streaming over HTTP in Content Centric Networks", in Proceedings of the IEEE International Conference on Multimedia and Expo 2013, San Jose, USA, July, 2013.

[13] Van Jacobson's talk at PARC Forum: "The Good, Bad, and Ugly of Digital Distribution: A Content-centric Networking Perspective on Evolving Network Architecture", February 2011.

[14] Hui Zhang, Li Tang and Jun Li, Impact of Overlay Routing on End- to-End Delay, Proceedings of 15th International Conference on Computer Communications and Networks, 2006. ICCCN 2006.

[15] GPAC, Multimedia Open Source Project. [Internet]
http://gpac.wp.mines-telecom.fr/

[16] *MP4Box,* General Documentation. GPAC, Multimedia Open Source Project. [Internet]
http://gpac.wp.mines-telecom.fr/mp4box/mp4box-documentation/

[17] Osmo4, *DASH Features in GPAC Players*, GPAC, Multimedia Open Source Project. [Internet]
http://gpac.wp.mines-telecom.fr/player/features/dash/

[18] *Osmo4,* How to build GPAC for Android. [Internet]
https://github.com/maki-rxrz/gpac/tree/master/build/android

[19] Adobe Systems Inc. Real-Time Messaging Protocol (RTMP) specification. 2009. [Internet]
http://www.adobe.com/devnet/rtmp.html

[20] Android developers' site. *Getting started* [Internet]
http://developer.android.com/training/index.html

[21] Android developers' site. *Application Fundamentals* [Internet]
http://developer.android.com/guide/components/fundamentals.html

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

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

[24] Sodagar, I., The MPEG-DASH Standard for Multimedia Streaming Over the Internet, IEEE Multimedia, vol. 18, no. 4, pp. 62-67, Oct.-Dec. 2011

[25] El uso de YouTube en cifras. [Internet]
http://blog.posicionamientoconvideomarketing.com/el-uso-de-youtube-en-cifras-una-vez-mas/

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

[27] Real Time Messaging Protocol. [Internet]
http://en.wikipedia.org/wiki/Real_Time_Messaging_Protocol

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

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

[30] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "*Networking named content*," in Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies. New York, NY, USA: ACM, December 2009

[31] *Project CCNx.* PARC, a Xerox Company. [Internet]
http://www.parc.com/work/focus-area/content-centric-networking/

[32] Networking Named Content. PARC. [Internet]
http://pages.cs.wisc.edu/~akella/CS838/F09/838-Papers/ccn.pdf

[33] *A dive into the caching performance of Content Centric Networking*. Giuseppe Rossini, Dario Rossi. Telecom ParisTech, Paris, France.

[34] P. Rodriguez, S.-M. Tan, and C. Gkantsidis, "*On the Feasibility of Commercial, Legal P2P content Distribution*," ACM SIGCOMM CCR, 36(1), 2006.

[35] D. Rayburn, "*CDN Market Getting Crowded: Now Tracking 28 Providers In The Industry*," Sep. 24th, 2007.

[36] IBC 2013 Conference. [Internet]
http://worg/page.cfm/Action=VCalendarIndex

[37] RTP Control Protocol (RTCP). [Internet]
http://es.wikipedia.org/wiki/Real_time_control_protocol

[38] DASH libraries. DASH Industry Forum. [Internet]
http://dashif.org/software/

Parte 8
Anexos

En esta parte de la tesis se procede a completar parte de la información anteriormente dada. Se procederá a explicar los pasos seguidos en el rooteo del terminal, los pasos seguidos para montar la red virtual CCN y el presupuesto estimado del proyecto.

I Root Android

Para el correcto funcionamiento del reproductor utilizado para las pruebas se necesitaba una versión de Android superior a la que inicialmente disponía el dispositivo móvil.

Preparación del entorno.

El primer paso es preparar el ordenador con el que se trabajará para realizar el root. Todos los siguientes pasos están pensados para realizarnos en un sistema operativo Windows.

El primer paso para preparar nuestro entorno de trabajo es disponer de la herramienta ***HTCSync*** y posteriormente instalar todos los drivers necesarios para el correcto funcionamiento de este programa con nuestro terminal. Una vez realizado este paso se procederá a la instalación del SDK en nuestro PC, aunque si ya se siguieron los pasos de configuración de GPAC ya se dispondrá de él [3.II.1]. Si no se dispone del SDK se explicará más detalladamente en esta sección.

- *Instalación y configuración del SDK.*

Antes de comenzar con la instalación del kit de desarrollo de herramientas para Android, hay que asegurarse que se cumplen con ciertos requisitos mínimos en nuestro equipo de trabajo.

- Windows XP (32-bit), Vista (32- bit o 64-bit), o Windows 7 (32-bit o 64-bit).
- Mac OS X 10.5.8 o posteriores solo (solo x86).
- Linux (Ubuntu Linux, Lucid Lynx)
  + GNU C Library (glibc) 2.7 o posteriores.
  + En Ubuntu Linux, versión 8.04 o posteriores
  + Las distribuciones de 64-bit deben ser capaz de ejecutar aplicaciones de 32-bit.

Por otro lado será necesario instalar un entorno de desarrollo, IDE (Integrated Development Environment). En nuestro caso Eclipse, que viene dado en las especificaciones de configuración de GPAC antes nombrada.

Una vez se tienen claras los requisitos mínimos necesarios para la instalación del kit de desarrollo, se procede a su descarga en la página de Android. Una vez se tiene este paquete ya se puede proceder a la instalación de los paquetes necesarios para el entorno de desarrollo mediante la herramienta Android SDK Manager. Para utilizar el Android SDK Manager se puede utilizar tanto en Eclipse como directamente accediendo en la carpeta correspondiente y ejecutándolo. Como en nuestro caso usamos Eclipse para el desarrollo de la aplicación y puesto a que es más simple utilizaremos este método. Para ello habrá que dirigirse a *Tools/Android SDK Manager* y aparecerá una ventana similar a la de la figura 29. En dicha ventana podremos configurar que paquetes instalar.

Una vez ya se tiene configurado e instalado el SDK se puede proceder al siguiente paso.

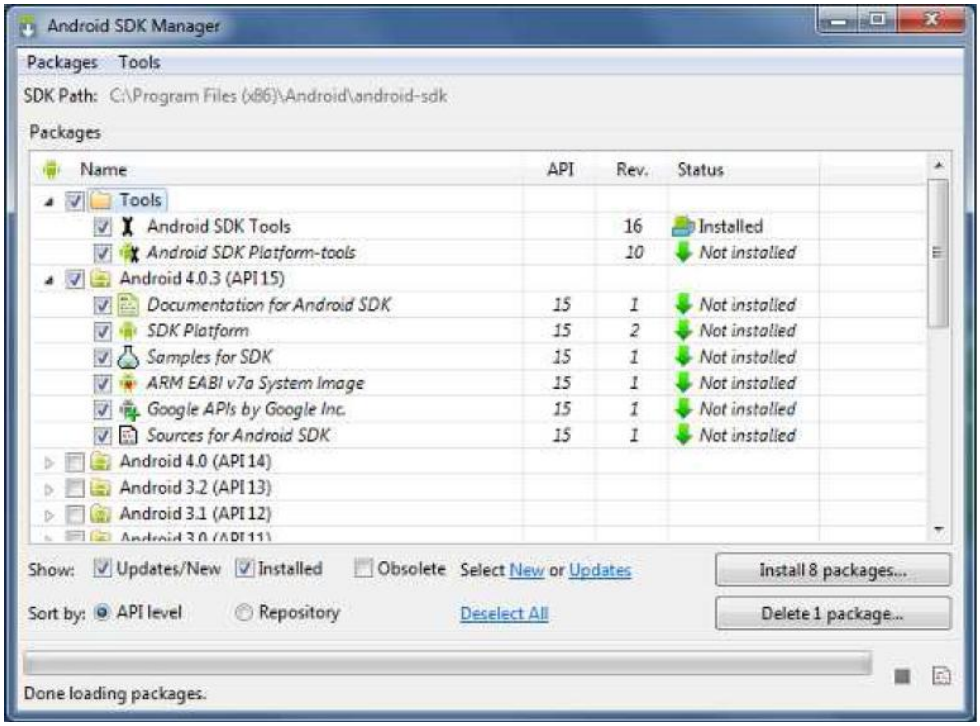


Figura 29 – Herramienta SDK Manager.

- *Creación de una goldcard.*

La *goldcard* es una tarjeta microSD modificada para permitir evadir las restricciones de bloqueo CID (código que identifica al proveedor del terminal y que define el nivel de protección cuando se accede desde un PC) en el terminal. Para convertir nuestra tarjeta microSD en una *goldcard* se seguirán los siguientes pasos.

En primer lugar nos debemos de descargar la siguiente herramienta:
http://www.filefactory.com/file/b1d16ag/n/GoldCardTool_exe

Una vez realizado esto, se debe desmontar la tarjeta microSD y formatearla. Se abre el ejecutable descargado ejecutándola como **Administrador**. Se mostrará una pantalla como la mostrada en la *Figura 30.*

Se conecta el terminal móvil al ordenador en modo "Sólo carga" y se selecciona "MMC0" y seleccionamos "Get CID". Si todo ha salido correctamente aparecerá al lado de esta casilla un código. Si no aparece se procederá a elegir "MMC1".

Abrimos la página de REVSKILLS de creación de Gold Card (Podemos hacerlo pinchando en la herramienta). En dicha página, se pone el código generado en el paso anterior, nos aseguramos que queremos una Gold Card Android, rellenamos el captcha y pinchamos en "Download Goldcard!". Al cabo de unos segundos, se permitirá descargar un archivo GoldCard.img que se guardará en la misma carpeta que la herramienta "Gold Card Tools".

A continuación se procede a poner el terminal en modo USB. En el programa se selecciona el botón “Refresh” y nos aparecerá señalada nuestra tarjeta microSD. Una vez esto, se procede a pinchar en “Load GoldCard.img” y seleccionaremos el archivo descargado. Se pincha en “Patch MMC” y se creará nuestra GoldCard.

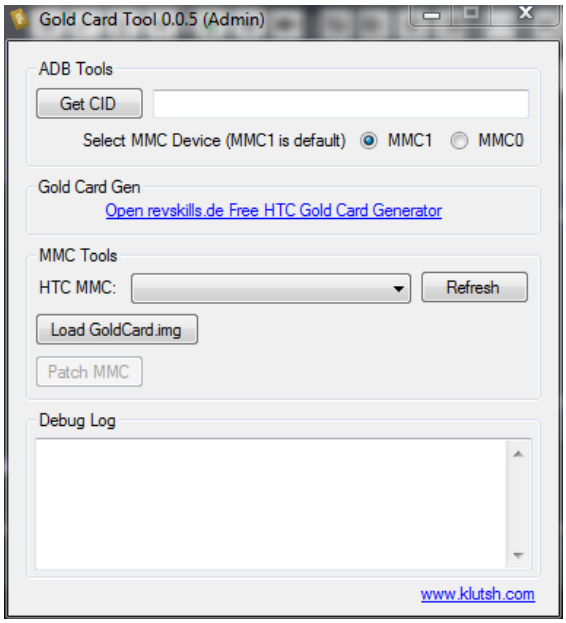


Figura 30 – GoldCard Tool.

Root terminal HTC.

Apagamos nuestro terminal HTC y lo encendemos manteniendo una combinación de teclas, botón de encendido + bajar volumen. Se mantiene pulsado hasta que aparezca una nueva pantalla donde seleccionaremos con la tecla de encendido (teclas de volumen para moverse por el menú) la opción FASTBOOT. En este momento se conecta el terminal al ordenador, aparecerá un mensaje “FASTBOOD USB”.

Se procede a conectar el terminal por USB al PC y abrimos el archivo *step1-windows.bat* de las herramientas que hemos descargado. Esperaremos a que se complete todo el proceso. En nuestro terminal la pantalla cambiará según vaya ejecutándose el programa. Una vez este proceso haya terminado, se volverá a la pantalla inicial “FASTBOOD USB”. En esta pantalla se selecciona la opción “BOOTLOADER” y posteriormente a “RECOVERY”, en este momento nos aparecerá una imagen con una señal de alerta. Ejecutamos *step2-windows.bat* y esperamos a que termine el proceso.

Posteriormente seleccionamos las opciones "Wipe"->"Wipe Data/Factory Reset" y cuando termine “Flash zip from SDCARD" y luego el archivo de que tengamos para el rooteo del terminal (con ROM cocinada o no). Por último, seleccionaremos “Reboot system now” y se reiniciará nuestro terminal rooteado y con la ROM seleccionada.

II Montar red virtual CCNx

Una de las partes más importantes en el desarrollo del proyecto es el correcto despliegue de la red utilizada para ejercer las pruebas concretadas. Se ha implementado una red CCN basado en el proyecto CCNx, antes de hablar de los detalles de la implementación se va a introducir algunos nombres propios del proyecto CCNx.

- *ccnd,* es un demonio CCN con propósitos de enrutamiento. Cada nodo CCN correrá su propio ccnd.

- *ccn-repo,* es una aplicación basada en CCN, su funcionamiento es ser un repositorio que permite el almacenamiento de datos CCN.

Con estos dos fundamentos claros, se procede a explicar los pasos seguidos para el despliegue de la red. Dependiendo del tipo de prueba se realizará un despliegue específico, ya que dependemos de la arquitectura. Para las pruebas sin la red CCN claramente no haría falta esta configuración.

Lo primero de todo es encontrarnos en el nodo que queramos mediante una conexión *ssh.* Dentro de cada nodo se procederá a la configuración de los mismos para su correcto funcionamiento.

Para las pruebas en las que disponemos de la arquitectura expuesta en [4.I.2] se realizará el paso 1, para las otras pruebas este paso se omite.

1. Una vez se ha realizado el paso anterior se procede a la configuración de la red. Lo primero es indicar en el fichero de configuración, del nodo 163.117.166.6, la IP de conexión de nuestro terminal móvil. Este archivo de configuración lo encontramos en nuestra red en la carpeta Dash/tc-5Mbps-CCN.click. Esto permitirá la conexión entre el nodo y nuestro terminal. Una vez cambiada la configuración se tendrá que proceder a la instalación del archivo, para ello se utilizará el siguiente comando:

   sudo click-install filename

2. A continuación se procede a la configuración del proxy HttpProxy. En HttpProxy se ejecutará el archivo *runProxy.sh* para configurar el funcionamiento del proxy y posteriormente se ejecutará el comando *./HttpProxy -resolveHigh ->proxy.log* con el que se arranca el proxy y se añade la información de la transmisión el fichero *proxy.log* creado.

3. El siguiente paso es la configuración del NetFetch. Para ello corremos CCNx con *ccndstart,* gracias a esto tendremos la configuración necesaria para realizar todo lo necesario para que el protocolo CCNx funcione correctamente. Una vez lanzada esta configuración, se puede borrar la caché del NetFetch escribiendo el comando: *rm -rf .hydra.netcom.it.uc3m.es/* , con esto nos aseguraremos de que se borra la información sobre el tráfico anterior. Finalmente se lanzará

./NetFetch -fsRoot -> fetch.log con el que arrancaremos el NetFetch y añadiremos al fichero *fetch.log* la información de la transmisión.

4. Por último se ejecutará el reproductor multimedia en nuestro terminal para la reproducción del vídeo demandado. Para ello sólo tendremos que indicar la URL donde se encuentra dicho vídeo, siempre solicitando el MPD. Por ejemplo:

http://163.117.166.10/sin.mpd

Parte 9
Planificación
y presupuesto

En esta parte de la tesis se procede a mostrar por un lado las fases del desarrollo del proyecto, y por otro lado el presupuesto desglosado.

I Planificación

En esta sección se va a mostrar la planificación seguida a la hora de comenzar a desarrollar el proyecto dividiéndolo en diferentes fases.

- **Fase 0. Documentación y estudio del arte.**

Esta primera fase tiene como objetivo la búsqueda de información para comenzar a comprender todas las técnicas necesarias para el correcto desarrollo y despliegue del proyecto. Se puede, a su vez, dividir en las siguientes subfases:

- *Subfase 0.1*: Estudio del funcionamiento del estándar DASH para vídeos multimedia. Conocer el funcionamiento del mismo y como se convierten los ficheros multimedia en dicho formato.

- *Subfase 0.2:* Estudio de las redes CCN, funcionamiento del protocolo CCNx y sus implementaciones.

- *Subfase 0.3:* Estudio de la integración de DASH sobre las redes CCN, comandos y posibles configuraciones a seguir para dicha integración.

- *Subfase 0.4:* Estudio de los paquetes necesarios para la correcta reproducción de archivos multimedia siguiendo el estándar DASH en el reproductor VLC.

- *Subfase 0.5:* Estudio de otras alternativas a VLC, GPAC.

- *Subfase 0.6:* Estudio de la plataforma Android para el cambio de configuración del reproductor Osmo4.

- **Fase 1. Configuración de Osmo4.**

Un esta fase se describen los comienzos en la elección del reproductor elegido y como se combina con DASH.

- *Subfase 1.1:* Elección del reproductor Osmo4 por delante de VLC gracias a su alta posibilidad de configuración y al ser un proyecto de software libre.

- *Subfase 1.2:* Estudio del entorno GPAC y su uso. Estudio de cómo adecuar el código de Osmo4 al proyecto para la representación de diferentes valores necesarios para la generación de las gráficas.

- *Subfase 1.3:* Descarga y configuración del reproductor Osmo4 para el correcto funcionamiento en terminales Android.

- *Subfase 1.4:* Root del terminal HTC Legend para cumplir con las especificaciones mínimas que se necesitan para Osmo4.

- *Subfase 1.5:* Estudio del comportamiento de Osmo4 en el dispositivo Android y configuración de las diferentes librerías para su funcionamiento.

- *Subfase 1.6:* Configuración de parámetros dentro del *.apk* para obtener datos importantes para el estudio de las pruebas (logs).

- **Fase 2. Pruebas con Emulador de Android.**

En esta fase se describen los primeros experimentos que se realizaron para probar el funcionamiento de las redes CCN.

- *Subfase 2.1:* Configuración de la red virtual CCN para la realización de las primeras pruebas.

- *Subfase 2.2:* Estudio sobre la captura de paquetes de información en el Emulador de Android.

- *Subfase 2.3:* Despliegue de la red virtual y pruebas con el emulador para redes sin CCN.

- *Subfase 2.4:* Despliegue de la red virtual y pruebas con el emulador para redes con CCN.

- *Subfase 2.5:* Despliegue de la red virtual y pruebas con el emulador para redes con CCN con arquitectura sin nodos intermediarios.

- *Subfase 2.6:* Estudio y conclusiones de los resultados obtenidos con el Emulador de Android.

- **Fase 3. Pruebas con HTC Legend.**

En esta fase se procede a realizar las pruebas con el HTC Legend, ya que como se ha visto, las realizadas con el Emulador de Android no fueron concluyentes.

- *Subfase 3.1:* Configuración de la red virtual CCN y pruebas con el dispositivo para redes sin CCN, en estático y con pérdidas de señal.

- *Subfase 3.2:* Configuración de la red virtual CCN y pruebas con el dispositivo para redes con CCN, en estático y con pérdidas de señal.

- *Subfase 3.3:* Configuración de la red virtual CCN y pruebas con el dispositivo para redes con CCN con arquitectura sin nodos intermediarios, en estático y con pérdidas de señal.

- *Subfase 3.4:* Estudio y conclusiones de los resultados obtenidos con el dispositivo móvil.

- **Fase 4. Memoria del proyecto.**

Una vez se tienen los resultados y conclusiones sobre el desarrollo del proyecto se procede a documentar el desarrollo y desenlace del mismo.

- *Subfase 4.1:* Redacción de la memoria.

- *Subfase 4.2:* Revisión y corrección de la memoria.

FASE	SUBFASE	DURACIÓN (semanas)
Fase 0: Documentación y estudio del arte.	0.1	2
	0.2	1
	0.3	2
	0.4	1
	0.5	1
	0.6	1
Fase 1. Configuración de Osmo4.	1.1	1
	1.2	2
	1.3	2
	1.4	1
	1.5	2
	1.6	4
Fase 2. Pruebas con Emulador de Android.	2.1	2
	2.2	1
	2.3	1
	2.4	1
	2.5	1
	2.6	1
Fase 3. Pruebas con HTC Legend.	3.1	1
	3.2	1
	3.3	1
	3.4	1
Fase 4. Memoria del proyecto.	4.1	6
	4.2	2
DURACIÓN TOTAL		**39**

39 semanas · 4 horas/día · 5 días/semana = 780 horas

Una vez contemplado la duración estimada del proyecto se va a proceder a analizar los recursos necesarios y utilizados para la realización de este proyecto.

- **Recursos materiales.**

- *Ordenador de sobremesa.* 1 PC Inter® Core® 2 Quad Processor Q940 @ 2.66GHz 4GB RAM. Sistema Operativo Debian 6 Squeeze.

- *Ordenador portátil.* 1 PC Intel® Core® i5 CPU @ 2.50 GHz, 4GB de RAM.

Sistema Operativo Windows 7 (64bits).

- *HTC Legend.* Android Jelly Bean 4.1(root), 384 MB RAM, 512 MB ROM memoria interna. Procesador Qualcomm MSM 7227 600 MHz

- **Recursos de trabajo.**

  - *1 Graduados en Ingeniería Telemática*. Para el desarrollo de la aplicación, despliegue de la arquitectura diseñada y gestión de las pruebas.

  - *1 Ingeniero Senior*. Para el control y dirección del proyecto.

- **Otros recursos.**

  - *Conexión a Internet*.
  - *Costes de mantenimiento de servidor web.*

II Presupuesto

En esta sección se detallen los costes de cada parte del proyecto para obtener el presupuesto total y los gastos que acarrearía la realización de éste.

1. Autor:

Daniel Martínez Ortiz

2. Departamento:

Ingeniería Telemática

3. Descripción del proyecto:

- ***Título:*** Reproductor de vídeo streaming sobre redes CCN.
- ***Duración:*** 1 año.
- ***Tasa de costes indirectos:*** 20%.

4. Presupuesto total del proyecto:

19.747,40€

5. Desglose presupuestario (costes directos):

PERSONAL			
Categoría	**Dedicación (hombres al mes)***	**Coste hombre hora bruto €**	**Coste €**
Ingeniero Telemático	780 horas	20	15.600,00
Ingeniero Senior	92 horas	30	2.760,00
Total			18.360,00

Coste = Coste hombre hora bruto (€) · Dedicación (horas)

* 1 Hombre mes= 131,25 horas. Máximo anual de dedicación de 12 hombres mes (1575 horas)
Máximo anual para PDI de la Universidad Carlos III de Madrid de 8,8 hombres mes (1155 horas)

EQUIPOS					
Descripción	**Coste €**	**%Uso dedicado proyecto**	**Dedicación (meses)**	**Período de depreciación**	**Coste imputable***
Ordenador sobremesa	600	100	12	60	360,00
Ordenador portátil	799	100	12	60	479,40
HTC Legend	90	100	12	120	108,00
Total					947,40

*Fórmula de cálculo de la amortización:

$$\frac{A}{B} x C x D$$

A = nº de meses desde la fecha de facturación en que el equipo es utilizado
B = periodo de depreciación (60 meses)
C = coste del equipo (sin IVA)
D = % del uso que se dedica al proyecto (habitualmente 100%)

OTROS					
Descripción	**Coste €**	**%Uso dedicado proyecto**	**Dedicación (meses)**	**Período de depreciación**	**Coste imputable***
Conexión Internet	40	-	12	-	40,00
Servidor Web	400	-	12	-	400,00
Total					440,00

TOTAL PROYECTO	**19.747,40**