UNIVERSIDAD POLITECNICA DE VALENCIA

ESCUELA POLITECNICA SUPERIOR DE GANDIA

Grado en Ing. Sist. de Telecom., Sonido e Imagen

"Diseño y desarrollo de una App que permita la reproducción de contenidos DVB en un dispositivo principal así como su filtrado, transmisión por IP y reproducción en dispositivos secundarios."

TRABAJO FINAL DE GRADO

Autor/a:

Juan Carlos Miñana Signes

Tutores:

Fernando Boronat Seguí
Daniel Marfil Reguero

GANDIA, a 10 de septiembre 2017

Contenido

Resumen.

Este Trabajo Final de Grado (TFG) consiste en el diseño e implementación de una solución que permite la transmisión y reproducción de contenidos DVB (*Digital Video Broadcasting*) recibidos por antena, en dispositivos IP dentro de la LAN del hogar. Los contenidos DVB son sintonizados por un servidor mediante una sintonizadora DVB-T (*Digital Video Bradcasting - Terrestre*), son procesados y puestos a disposición de dispositivos conectados a la red IP del hogar, que accederán a ellos mediante tecnología HTTP-Adaptive Streaming basada en HLS (HTTP Live Streaming) .

El primer usuario que se conecte podrá seleccionar en su dispositivo el programa a visualizar de un determinado canal televisivo de entre todos los que se reciben en el hogar. Además, las aplicaciones desarrolladas permiten que el mismo o diferente programa o elemento multimedia (p.ej., vídeo, audio en diferentes idiomas…) del mismo canal seleccionado por dicho usuario, se reproduzca en uno o varios dispositivos secundarios bajo demanda. Para ello, se han diseñado protocolos de descubrimiento, señalización y *streaming* adecuados.

En concreto, las aplicaciones desarrolladas en el TFG permiten los siguientes dos casos de uso:

a) En un dispositivo principal se selecciona y se visualiza un programa específico (p.ej., TVE) de TDT (Televisión Digital Terrestre) de un canal UHF, mientras que en un dispositivo secundario se puede visualizar bien el mismo contenido o bien otro programa incluido en el mismo múltiplex (p.ej., 24h) recibido en dicho canal.

b) Un dispositivo puede estar reproduciendo un programa específico en un idioma A, mientras que en los dispositivos secundarios se puede seleccionar el audio en otro idioma (de los disponibles en el múltiplex para ese programa). Este caso de uso contribuye tanto a una mejor integración social como a una mayor personalización de las experiencias de TV, además de facilitar el aprendizaje de idiomas.

La solución desarrollada ha sido evaluada por usuarios con resultados satisfactorios y prometedores.

Palabras clave: TDT, Gstreamer, Streaming, HLS, Android.

Resum.

Aquest Treball Fi de Grau (TFG) consisteix en el diseny i implementació d'una solució que permet la transmissió i reproducció de continguts DVB (*Digital Video Broadcasting*) rebuts per antena, en dispositius IP dins de la LAN de la llar. Els continguts DVB són sintonitzats per un servidor mitjançant una sintonitzadora DVB-T (*Digital Video Broadcasting - Terrestre*) i són processats i posats a la disposició de dispositius connectats a la red IP de la llar, que accediran a ells mitjançant tecnologia HTTP - Adaptive Streaming basada en HLS (*HTTP Live Streaming*).

El primer usuari que es connecte podrà seleccionar en el seu dispositiu el programa a visualitzar d'un determinat canal televisiu d'entre tots els que es reben en la llar. A més, les aplicacions desenvolupades permeten que el mateix o diferent programa o element multimedia (p. ex vídeo, audio en diferents idiomes...) del mateix canal seleccionat per aquest usuari, es reproduixca en un o diversos dispositius secundaris baix demanda. Amb aquesta finalitat s'han dissenyat protocols de descobriment, senyalització i streaming adequats.

En concret, les aplicacions desenvolupades en el TFG permeten els següents dos casos d'ús:

a) En un dispositiu principal es selecciona i es visualitza un programa específic (p. ex., TVE) de TDT (Televisió Digital Terrestre) d'un canal UHF, mentre que en un dispositiu secundari es pot visualitzar bé el mateix contingut o bé un altre programa inclòs en el mateix múltiplex (p.ex., Teledeporte) rebut en aquest canal.

b) Un dispositiu pot estar reproduint un programa específic en un idioma A, mentre que en els dispositius secundaris es pot seleccionar l'àudio en un altre idioma (dels disponibles en el múltiplex per a eixe programa). Aquest cas d'ús contribueix tant a una millor integració social com a una major personalització de les experiències de TV, a més de facilitar l'aprenentatge d'idiomes.

L'aplicació ha sigut evaluada per usuaris amb resultats satisfactoris.

Paraules clau: TDT, Gstreamer, Streaming, HLS, Android.

Abstract.

This Final Project consists of the design and implementation of a solution for transmitting and playing back DVB (Digital Video Broadcasting) contents, which are received via broadcast, to IP devices inside a home LAN. DVB contents are tuned in by a server with a DVB-T (DVB-Terrestrial) tuner card. Then, they are processed and left available for other IP-connected devices at home. The latter will access to these contents through HTTP-Adaptive Streaming technology, specifically HLS (HTTP Live Streaming).

The first user to connect to the server will be able to select in his/her own device the TV program to watch from all the available channels. Moreover, the developed applications allows that the same, or different multimedia elements (e.g., video, or audio for different languages…) from the chosen TV channel can be played in one or more secondary devices on demand. Discovery, signaling and streaming protocols have been used and designed for this purpose.

The applications which have been developed for this project consider the following two use cases:

a) A main device (user) selects and plays a specific TV program received through an UHF channel (e.g., TVE program, broadcasted by the Spanish public broadcaster). Afterwards, another device can select the same content, or content from a different TV program, but received through the same UHF channel (as long as it is part of the received multiplexed content), as, for example, 24h program.

b) A device is playing a specific program in A language and, meanwhile, in another device, a different audio language (if available) can be selected. This use case contributes to a better social integration of impaired people and to a more customizable experience of watching TV, even improving language learning.

The developed solution's performance has been tested by users with successful and promising results.

Key words: TDT, Gstreamer, Streaming, HLS, Android.

Agradecimientos.

En este apartado del trabajo quiero dar las gracias por el tiempo invertido, ayuda y en temas de asesoramiento a Fernando Boronat, Dani Marfil, Marc Martínez y Mario Montagud, pertenecientes al grupo de investigación IIM (Immersive Interactive Media R&D Group) del Departamento de Comunicaciones en el Campus de Gandía de la Univesitat Politècnica de València).

Y como no, también dar las gracias a mis padres, hermana, familia y pareja por el apoyo prestado y por haberme motivado a llegar dónde me he propuesto llegar, a la realización de este TFG y la finalización de mis estudios del Grado en Ingeniería de Sistemas de Telecomunicación, Sonido e Imagen

1.1. Introducción.

La televisión (TV, en adelante) es el gran medio de comunicación a través del cual se pueden enviar imágenes y audio en tiempo real, a través de las ondas que viajan por el aire o cualquier otro medio, todo esto gracias a la ayuda de las tecnologías existentes. Tal es su importancia en el ámbito doméstico, que se tienen registradas leyes sobre ICT (Infraestructuras Comunes de Telecomunicación) en edificios en las que se obliga a tener un mínimo de número de tomas de antenas en las viviendas para el uso de la TV y proporcionar acceso a las personas al servicio de TV.

Cabe mencionar, que no todas las familias pueden tener un televisor en cada estancia de la vivienda debido al alto coste que ello puede resultar. Además, un mismo televisor sólo puede ofrecer en tiempo real un sólo canal televisivo, ya que la gran mayoría de ellos solo contienen un sintonizador y un decodificador y no pueden mostrar dos canales al mismo tiempo. También, hay que recalcar que la sociedad de hoy en día, cada vez busca más intimidad y los jóvenes ya no suelen estar en la misma estancia que los padres a la hora de ver la TV, ya sea por no poder ver el programa deseado o no compartir los mismos gustos. En la imagen 1, se puede observar que, con el paso de la edad, las personas hacen más uso de la TV que cualquier otro medio de información.

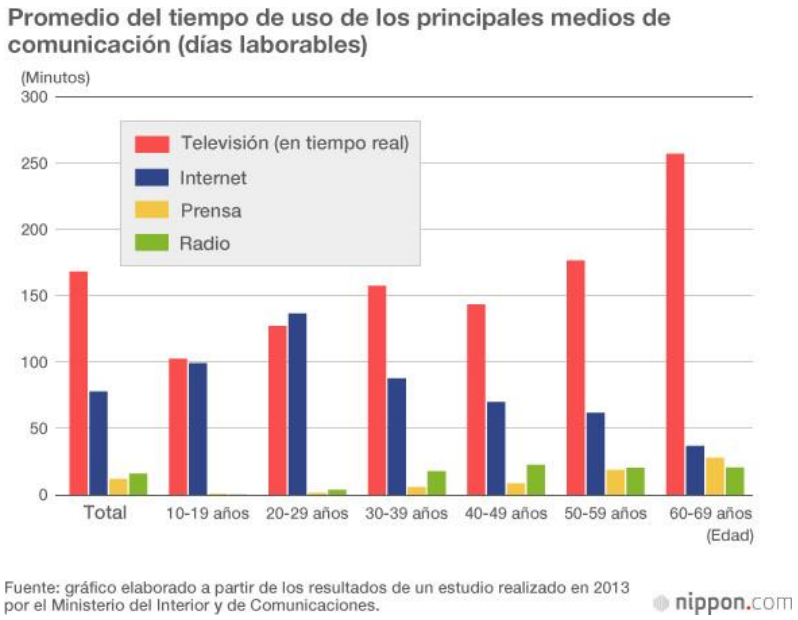

Imagen 1. Uso medio de soportes de comunicación diferenciado por edades

Como se ha comentado anteriormente, los jóvenes buscan cada vez más intimidad para poder tener acceso a estos medios. Esto conlleva a aumentar la peligrosidad de

tener acceso a material no autorizado sin estar bajo la supervisión de los padres o tutores, es decir, en la actualidad es muy fácil poder tener acceso a contenido multimedia violento o incluso contenido sexual, no apto para menores. En la imagen 2, se muestra el porcentaje de uso que emplean los menores españoles de TV e Internet, las dos mayores tecnologías de información que existen en la actualidad. Destacar que se empieza muy temprano a utilizar Internet, donde se ofrece una gran variedad de información.

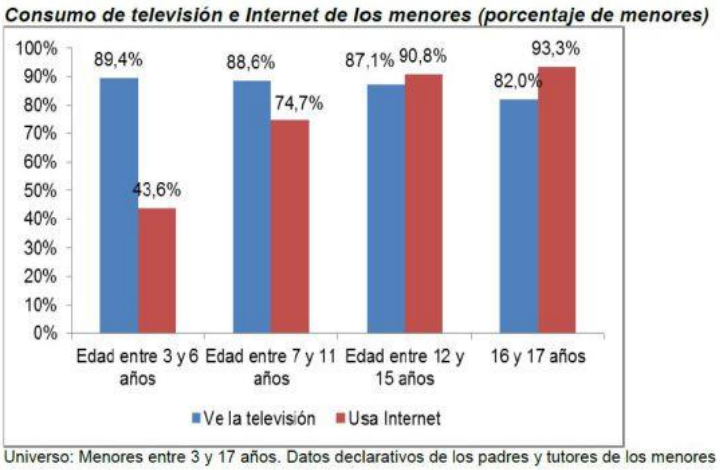

Imagen 2. Consumo en porcentaje que presenta el uso de TV e Internet en los jóvenes

Presentada esta problemática, en el siguiente sub-apartado se darán a conocer los objetivos del TFG para plantear una solución que pueda resolver el conflicto de poder dar la posibilidad de ver distintos contenidos que ofrece la TV sin la necesidad de poseer más de un televisor, sino utilizando dispositivos con pantalla con conexión IP (wifi).

Una vez comentados los problemas citados anteriormente, se ha realizado una búsqueda por Internet con la finalidad de encontrar trabajos relacionados (*Related Works*). En esta búsqueda se han encontrado servicios que resolverían este conflicto o se basan en la idea que se ha utilizado para hacer frente a la situación planteada.

Por un lado, el sintonizador digital de TV para Xbox ofrece la posibilidad de ver TV pero sólo en dispositivos que soporten el sistema operativo Windows. Para más información visitar el siguiente enlace proporcionado: *https://www.xataka.com/videojuegos/television-hasta-en-la-sopa-y-ahora-tambien-en-la-xbox-one*

Otro dispositivo que conviene mencionar es el *hdhomerun-connect* que contiene dos sintonizadores en su interior y da la posibilidad de ofrecer dos canales de TV totalmente distintos al mismo tiempo. Su desarrollo está pensado para el consumo doméstico. Su costo es de 151€ (distribuido por Amazon).

Para más información se recomienda la visita al siguiente enlace: *https://www.silicondust.com/product/hdhomerun-connect/*

Otro producto similar al anterior es el que ofrece Tablo. En su página oficial se puede elegir hasta un decodificador con 4 sintonizadores. El uso del producto va ligado a compatibilidades de sistemas donde se va a ejecutar. En el caso de no ser compatible por ejemplo con una televisión, contienen otro dispositivo que realiza un puente de conexión entre el decodificador y la TV. En el caso de la App desarrollada en este trabajo, al utilizar VLC solventamos ese problema de pasarela de comunicación.

https://www.tablotv.com/products/

1.2. Objetivos.

El principal objetivo de este trabajo es la creación y desarrollo de una solución que sea capaz de realizar la recepción de contenido proveniente de una señal de TDT (tecnología *broadcast*) y retransmitirlo a dispositivos secundarios mediante streaming IP (tecnología broadband), también conocidos como segundas pantallas (*secondary screen*), como pueden ser *smartphones*, *tablets* y PCs interconectados mediante un red local (normalmente, inalámbrica). Dicha solución no requiere de conexión a Internet, es decir, no supone coste económico por suscripción ni ancho de banda limitado. Sólo es necesario una red local o entorno wifi que incluso lo podría proporcionar uno de los dispositivos involucrados.

Con este objetivo, se pretende poder oír o visualizar uno o más programas de TDT distintos dentro del mismo multiplex del canal sintonizado en cada momento. Para ello, será necesario conocer la codificación que emplea cada radiodifusor TDT y desarrollar un módulo servidor que será el encargado de decodificar/codificar la señal y retransmitirla a todo dispositivo compatible que se encuentre en la misma red. Este servidor, realizará el procesado de información para permitir su disponibilidad en la red local donde esté conectado.

Cuando el primer dispositivo secundario seleccione un programa de TV para visualizar, dentro de un múltiplex de TDT, se sintonizará el canal correspondiente a dicho múltiplex. A partir de dicho momento, todos los demás dispositivos (usuarios) que se conecten posteriormente, sólo podrán visualizar programas pertenecientes al múltiplex recibido por el canal que previamente haya elegido el primer usuario. Esto es debido a que sólo se dispone de una tarjeta demoduladora (es decir, un único sintonizador). Para los dispositivos secundarios (clientes), se ha desarrollado una aplicación para dispositivos Android, realizada con Android Studio.

Con la construcción de esta App, aparte de ofrecerle al usuario otra forma de ver contenidos de TV, se pretende conseguir una mejor relación social en un entorno dónde se esté reproduciendo material audiovisual y estén presentes distintas personas que hablen diferentes idiomas. La aplicación permite en la medida de lo posible (si el programa incluye audio en varios idiomas, o si el dispositivo dónde está ejecutándose tiene suficientes recursos para ello) poder recibir información audiovisual según los requisitos que elija el cliente.

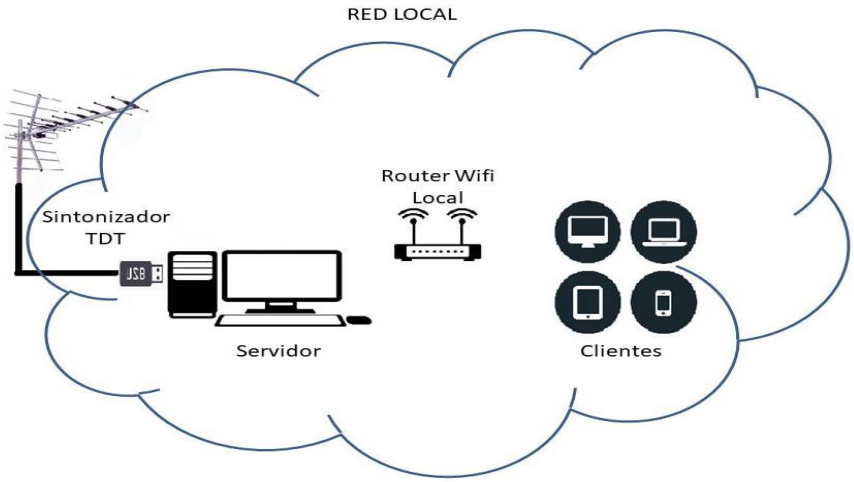

Imagen 3: Escenario de uso de la App. Se observan elementos como la antena, sintonizador TDT, router wifi con conexión a Internet y dispositivos secundarios como pueden ser tabletas, smartphones o Pc (Fuente propia).

En la figura anterior se puede observar el escenario de la solución desarrollada. Como elemento principal es necesario el uso de un decodificador TDT para poder captar la señal audiovisual que esté disponible a través de la antena. Este decodificador estará conectado a un pc para su posterior procesamiento, o bien, conectado a otro dispositivo como un mini PC (PC de bajo coste con recursos limitados). Estos dispositivos se encargarán de reproducir la información seleccionada a otros dispositivos secundarios conectados en la misma red a través de wifi.

Para simplificar, el primer usuario en conectarse a la App, será el encargado de seleccionar un programa para su visualización. De manera transparente al usuario, se filtrarán todos los canales que compartan ese mismo multiplex seleccionado para que un segundo o posterior usuario a la aplicación, tenga la oportunidad de visualizar el mismo contenido u otro perteneciente al mismo rango de emisión que el primer elemento elegido. Esto se debe a que el receptor DVB-T estará ya siendo utilizando para una determinada frecuencia de canal (el del múltiplex seleccionado) y por tanto un segundo usuario no puede cambiarlo, ya que interferiría en la experiencia del

primer usuario. Es por ello que a partir del segundo usuario conectado la oferta de canales que pueden ser visualizados corresponderá únicamente a los pertenecientes al múltiplex cuya frecuencia esté en uso.

Para una mejor aclaración se describen por apartados los objetivos perseguidos.

1. Crear de una App que permita seleccionar un flujo multimedia de tipo DVB-T y que haga disponible su contenido transportándolo a través de un entorno de red. Esta es la parte dedicada al servidor. Para su implementación se deberá cumplir los siguientes sub-objetivos:

   - Configurar el receptor de señal TDT.
   - Generar un fichero para almacenar toda la información decodificada por el receptor para su posterior análisis.
   - Desarrollar el funcionamiento del servidor para la posterior retransmisión de material multimedia. Aquí se incluye la necesidad de uso de protocolos como DIAL (*Discovery And Launch*), o tecnología como HLS (*HTTP Live Streaming*).

2. Crear una App cliente que permita seleccionar y recibir contenido multimedia a través de la información que ofrece la parte del servidor. Este apartado va relacionado con el cliente. Para ello se necesita:

   - Crear una interfaz gráfica que actúe de forma dinámica.

Desarrollar una App (mensajes utilizados para la comunicación entre servidor y cliente, configuración de la recepción del material multimedia recibido) para el usuario donde pueda controlar el flujo que quiere reproducir en su dispositivo.

1.3. Estructura de la memoria y documentación anexa.

La memoria está estructurada de la siguiente manera:

En el capítulo 2 está expuesto todo el material necesario para desarrollo final de este trabajo, tanto hardware como software.

A continuación, en el capítulo 3, se exponen todas las tecnologías empleadas en el TFG.

Seguidamente, en el capítulo 4 se explica detalladamente todo el proceso de creación y desarrollo de los dos módulos de la aplicación, paso a paso, desde la definición del framework Gstreamer, así como el uso de scripts desde terminal para agilizar el conocimiento de uso de elementos derivados a Gstreamer y un esquema que hace más descriptivo el ejemplo mostrado. También se definirá qué papel ha tenido la herramienta software *DVB inspector* en la realización del proyecto. Además, se incluye la explicación de la implementación de los dos módulos desarrollados tanto para la parte del servidor como para la del cliente.

En el capítulo 5, se exponen las conclusiones y algunas valoraciones subjetivas sobre el impacto que puede tener la aplicación de esta App en la vida cotidiana.

La memoria finaliza con el apartado de referencias bibliográficas, dónde se puede observar las fuentes consultadas para obtener la información necesaria para poder realizar este TFG.

Aparte de este documento, en la documentación del TFG se adjuntan los siguientes anexos complementarios a la memoria:

Anexo I: Puesta a punto del equipo.

Anexo II: Flujos de programa y de transporte de MPEG2.

Capítulo 2. Herramientas.

En este apartado se presentan brevemente aquellas herramientas *hardware* y *software* que ha sido necesario utilizar para la realización del TFG. Las herramientas *hardware* son elementos físicos que constituyen un ordenador o que le aportan funcionalidades. Por otra parte, el *software* es el mecanismo que se encarga de controlar el funcionamiento del *hardware*. Una vez diferenciados, el paso siguiente es poner en conocimiento las herramientas empleadas para el desarrollo de este trabajo.

2.1. Herramientas Hardware.

En el TFG se hace uso de los siguientes dispositivos hardware:

a) Ordenador (PC), marca MSI, modelo DC111-027XEU. Este equipo se ha utilizado para la instalación, creación y desarrollo de la aplicación como de todos sus componentes de programación. Se destacan las siguientes características del PC.

- Memoria RAM de 4 GiB.
- Procesador Intel Celeron a 1.8 GHz.
- Tarjeta gráfica Intel HD
- Sistema operativo de 64 bits.

b) Tarjeta capturadora de TV (DVB-T) con conectividad USB, marca Sveon, modelo STV21, para poder captar la señal directamente de la toma de antena de recepción de TV, para su posterior procesado.

Imagen 4. Sintonizadora TDT Sveon STV21 con conexión USB.

c) Dispositivo móvil Samsung S5 y Tablet Samsung Galaxy Tab S con sistema operativo Android

2.2. Herramientas Software.

En el TFG se han empleado las siguientes herramientas software:

a) S.O. Linux Ubuntu, versión 14.04 LTS de 64 bits. Se decide realizar todo el proceso de construcción de la App en el entorno del sistema operativo Linux Ubuntu ya que pertenece a software libre y está totalmente apartado de cualquier tipo de copyright.

c) IDE (entorno integrado de desarrollo) Eclipse. Este entorno se ha utilizado para la implementación del módulo del servidor.

d) IDE Android Studio, se ha empleado para el desarrollo de la parte del cliente, puesto que solo se implementa para dispositivos que operen con Android.

e) Framework Gstreamer versión 1.6.4. Herramienta principal del trabajo, ya que es la que permite el manejo del material audiovisual. Se hace uso de ella en Eclipse, parte del servidor. Gstreamer es un framework multimedia libre multiplataforma desarrollado en lenguaje de programación C empleando el uso de la librería GObject. Su función consiste en poder ofrecer un flujo de datos y manejo o negociación entre distintos tipos de medios. Además provee una API para crear las aplicaciones.
En principio esta herramienta es de uso libre pero, para su posterior comercialización de la App, hay que indicar que se va emplear dicha herramienta a la organización que conforma Gstreamer, ya que se hace uso de este framework.

Para profundizar un poco más en Gstreamer, se procede a exponer sus conceptos básicos para un mejor entendimiento.

Elements: son los elementos fundamentales pertenecientes de la clase de objetos en Gstreamer. Estos elementos permiten crear cadenas de elementos enlazados entre ellos para poder lograr una fluidez de datos, es decir, un elemento tiene funciones específicas como bien pueden ser, leer datos de un archivo, decodificarlos o incluso encargarse de la transmisión de ellos a una tarjeta de sonido, de vídeo u otro dispositivo.

Situando varios elementos se puede llegar a realizar distintas tareas como puede ser reproducción o captura multimedia. Gstreamer posee una amplia posibilidad de elementos, además, si es necesario, cabe la posibilidad de introducir elementos intermedios para el desarrollo de complementos.

Bins: este concepto recibe la propiedad de contenedor de elementos. Es considerado como subclase de los elements. Su función es la de cambiar el estado de todos los elementos de un bin solo cambiando el estado del bin contenedor.

Pipelines: están considerados como bins pero de mayor nivel o subtipos de bin.

Pads: este tipo de elementos se utilizan para negociar enlaces y flujo de datos entre los distintos elements de Gstreamer. Existen de dos tipos, source (de salida) y sink (de entrada).

Se profundiza más este apartado en el capítulo 3 referente a las tecnologías empleadas en el TFG.

f) La herramienta DVB inspector, permite examinar en detalle cada paquete de un archivo con formato (extensión) TS (MPEG-2 Transport Stream). Este programa ayuda a encontrar que tipo de codificación contiene cada programa que se emite en un rango de emisión. Cabe destacar que, gracias a este programa, se extraerán datos importantes, como pueden ser el *program-number* y los PID (Identificador de Paquetes) de audio y vídeo, de los que, posteriormente, se hará uso para verificar de forma de testeo que la pipeline de emisión está emitiendo el material escogido. La siguiente imagen, muestra el contenido obtenido de un archivo .TS mediante esta herramienta para poder identificar que paquetes corresponden a cada elemento multimedia, ya sea audio o vídeo.

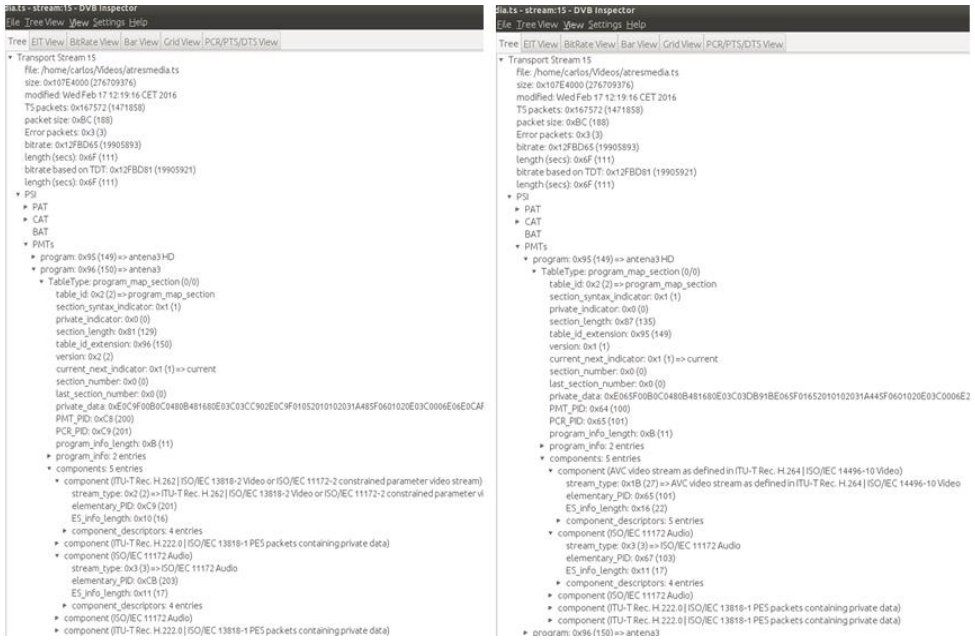

Imagen 5. Tablas de información extraídas con DVB Tools referentes al canal Antena3.

Con formato: Izquierda, Sangría: Primera línea: 0 cm, Espacio Después: 10 pto, Agregar espacio entre párrafos del mismo estilo

g) DVB tools. Este software permite el escaneo del espectro correspondiente a la red de TDT de España. Al iniciarse, crea un archivo con extensión *.conf* que contiene una lista con información sobre cada canal encontrado. Este archivo es generado mediante el uso de un script desde un terminal. El script tendría la siguiente forma:

```
scan -o zap /usr/share/dvb/dvb-t/es-Valencia > channels.conf
```

El resultado sería como el mostrado en la imagen 6.

```
tdp:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:2001:2002:40002
tdp HD:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:4001:0:40003
Radio Clasica HQ:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:2010:40005
Radio 3 HQ:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:3010:40006
Radio Exterior RNE:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:2511:40007
Canal Ingenieria:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:0:40010
9Kiss TV:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1001:1002:41001
Kiss FM:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:1101:41002
10:482000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:501:502:41011
antena3 HD:626000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:101:103:149
antena3:626000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:201:203:150
laSexta HD:626000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:301:303:151
laSexta:626000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:401:403:152
neox:626000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:501:503:153
nova:626000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:601:603:154
Telecinco:650000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1101:1103:186
Cuatro:650000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1201:1203:187
FDF:650000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1301:1303:188
Divinity:650000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1401:1403:189
Telecinco HD:650000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1501:1503:190
Cuatro HD:650000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_AUTO:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1701:1703:191
La 1:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_1_2:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:101:103:570
La 2:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_1_2:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:201:203:571
24h:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_1_2:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1001:1003:572
Clan:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_1_2:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1501:1503:573
La 1 HD.:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_1_2:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:301:0:574
Radio Nacional:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_1_2:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:2001:575
Radio 5:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_1_2:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:0:2031:576
```

Imagen 6: Formato de creación de un archivo con extensión *.conf*, generado con DVB tools.

Se puede observar que de cada canal encontrado se obtiene su nombre, frecuencia empleada, ancho de banda, modulación de amplitud en cuadratura(QAM), modo de transmisión, intervalo de guarda, número de PID de audio, número de PID de vídeo y program-number. Cada elemento citado anteriormente va separado entre dos puntos. Para obtener más información acerca de su instalación en Ubuntu se puede consultar en el Anexo

Capítulo 3.Tecnologías empleadas.

A continuación, en este capítulo se presentan las tecnologías necesarias para la realización de este trabajo. Ellas son MPEG2 (*Moving Pictures Experts Group 2*), DIAL, uso de Socket UDP (*User Datagram Protocol*) y HLS.

3.1. Tecnología MPEG2

La señal de material de entrada y de salida audiovisual se denomina *MPEG-2 Transport Stream* (TS) o flujo de transporte. Para emplearse en DVB, se debe complementar haciendo uso de *Información del Servicio* (SI). Se multiplexan varios programas audiovisuales formando un múltiplex donde cada programa está compuesto por uno o varios flujos elementales (ES) paquetizados (PES) y encapsulados en paquetes de transporte (transport packets). Cada uno de estos paquetes, al mismo tiempo, está marcado con PIDS que indica a qué tipo de flujo elemental pertenecen.

Para una mejor aclaración, un Programa, empleando la tecnología MPEG, es un servicio o canal simple de radiodifusión que cuando se comprime, cada componente simple del programa recibe el nombre de Elementary Stream (ES). Un programa puede estar compuesto por varios ES (vídeo, audio, subtítulos, teletexto…). Cada ES se estructura en paquetes que reciben el nombre de Packetised Elementary Stream (PES), es decir, por cada PES habrá un ES original.

Para que un decodificador pueda recuperar por completo un programa a través de la ayuda que le ofrecen los PIDS, es necesario incluir información adicional dentro del flujo de transporte para que relacione los valores de PIDS con los programas a los que pertenecen. Esta información recibe el nombre de Program Service Information (PSI) o Información Específica de los Programas.

La tabla que representa PSI definida por MPEG-2 está compuesta por cuatro tipos:

- PAT (*Program Association Table*). Tabla de inclusión obligatoria que contiene lista de todos los programas disponibles en el TS.
- CAT (*Condicional Acces Table*).Esta tabla aparecerá únicamente si algún canal del multiplex es de acceso condicional. Esta información no está especificada en MPEG-2 ya que depende del tipo de sistema de cifrado que se haya empleado.
- PMT (*Program Map Table*). Esta tabla proporciona información acerca del programa y de los flujos que se asocian a él.
- NIT (*Network Information Table*). Incluye información de red.

También hay que mencionar *MPEG-2 Program Stream* (PS) o también denominado flujo de programa, ya que este flujo de señal se usa para almacenar y recuperar

información digital en entornos libres de cualquier error. A diferencia de TS, este sólo puede multiplexar un solo programa.

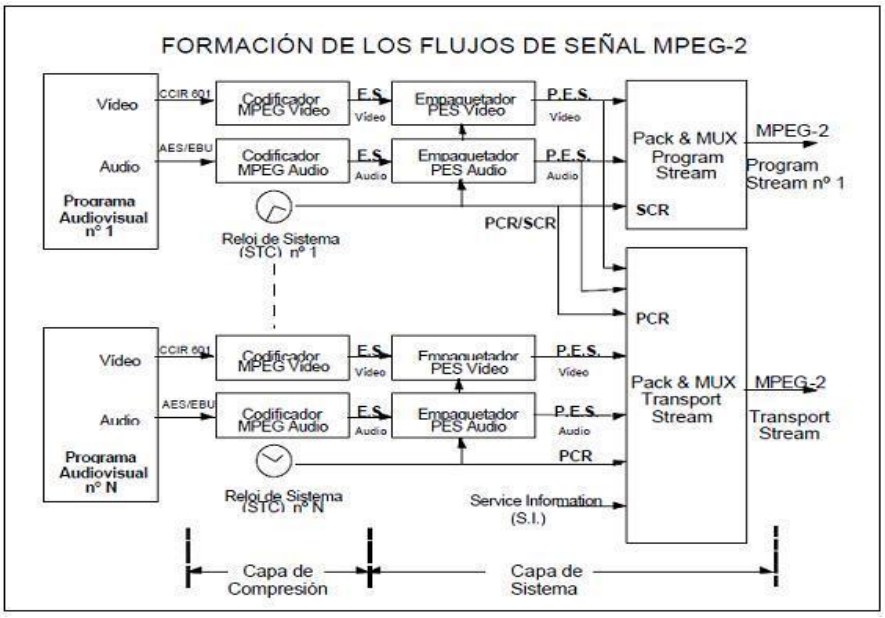

Imagen 7: Esquema resultante para la formación de flujos de señal MPEG-2. [3]

En el caso de PS, se multiplexan el audio, vídeo, datos, etc… Y se le añade un reloj del sistema. Sólo se transmite la información perteneciente a un único programa en el que se debe de compartir, de forma obligatoria, un mismo reloj de referencia.

El PS está formado por PACKS que a su vez están formados por cabecera del pack (pack-header), otra opcional (system-header) y numerosos PES-packets pertenecientes a los ES del programa audiovisual. Estos packs no tienen definida una longitud de paquete.

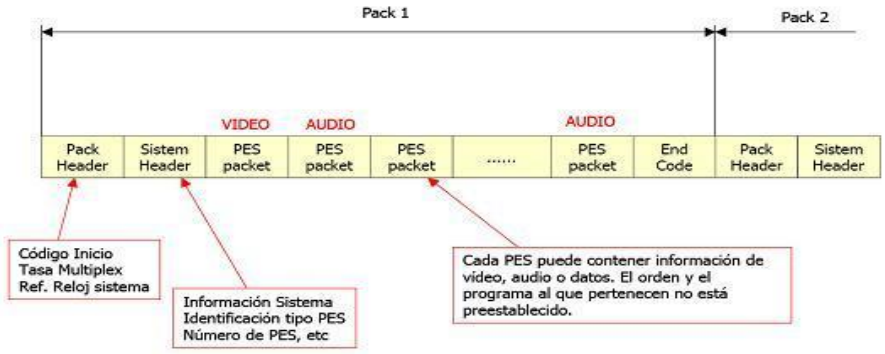

Imagen 8: Estructura que conforma un pack de Program Stream. [4]

Por otro lado, un múltiplex TS contiene varios programas (PS) y, además, incluye más información relacionada con el servicio, como pueden ser la Tabla de Asociación de Programas(PAT), la Tabla con Información para Acceso Condicional(CAT), y la Tabla de Mapa de cada Programa(PMT)…

En la siguiente figura, se puede observar que cada programa puede contener video, audio e información incrustados con marcas temporales. Todos estos programas se juntan formando el multiplex de emisión para que con una frecuencia de emisión se puedan enviar varios canales de TV al mismo tiempo.

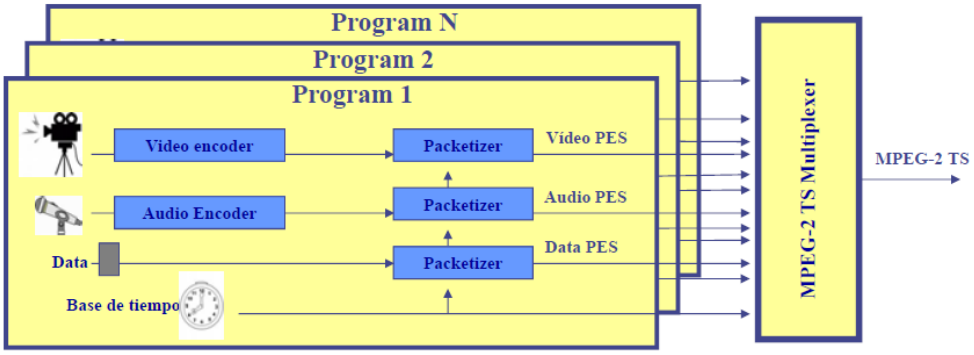

Imagen 9: Programas de TV que comparten un mismo multiplex MPEG2.

Cada paquete de transporte de TS siempre tiene una longitud fija de 188 bytes (cabecera de 4 bytes a veces seguida de otro campo denominado de adaptación, que en caso de no estar lleno se rellena el exceso de espacio disponible, y finalmente un campo llamado de carga útil o Payload). En la siguiente imagen se muestra detalladamente la estructura que guarda un TS.

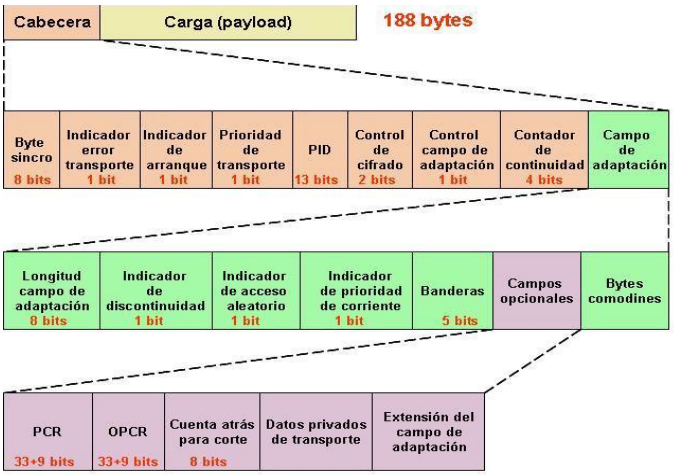

Imagen 10: Estructura de un paquete de Transport Stream. [5]

3.2. tecnología ***DIAL*** (*Discovery and Launch*)

DIAL es un protocolo que se emplea para poder descubrir y lanzar aplicaciones dentro de una subred, es decir, dentro de un entorno de red doméstico. Se basa en protocolos como son UPnP (*Universal Plug and Play* ,conjunto de protocolos de comunicación que permite a periféricos en red vincularse de forma trasparente a los usuarios para establecer servicios de red, entretenimiento y compartición de archivos), SSDP (*Simple Service Discovery Protocol* es un protocolo que sirve para la búsqueda de dispositivos UPnP en una red) y HTTP (*Hypertext Transfer Protocol*, protocolo de comunicación que permite las transferencias de información en la World Wide Web). Su creación nace a partir de la necesidad de conectar dispositivos sin previamente emparejarlos de forma visible a los usuarios. La conexión es de forma transparente al cliente. Se empieza a utilizar en los dispositivos de *chromecast* de Google y sus desarrolladores son Netflix y Youtube con colaboraciones de los fabricantes de Sony y Samsung. Tiene dos componentes, una de descubrimiento que permite encontrar servidores DIAL en su entorno de red, y el servicio DIAL rest, que se encarga a través del cliente iniciar, consultar o incluso detener aplicaciones que este ejecutando un servidor DIAL. Para más información, se puede consultar en el Anexo.

Con formato: Inglés (Estados Unidos)

Con formato: Sangría: Primera línea: 0 cm

3.3. ***Socket UDP*** (*User Datagram Protocol*).

El uso de sockets es considerado como un mecanismo encargado para la transmisión de datos a través de la red. Este queda definido una vez tiene definidas dos direcciones de red, la local y remota, al igual que sus números de puerto, aparte de llevar incluido el protocolo de transporte UDP. Permite establecer un orden entre cliente y servidor. Para ello debe de estar definido tanto en la parte que lo inicia, el

cliente, como en el servidor para así poder realizar una comunicación fiable. Al contemplar un protocolo no orientado a la conexión (UDP) no se garantiza que lleguen al cliente todos los mensajes en orden de envío o incluso que ni lleguen, lo que si se asegura es que si llega un mensaje, este llegará bien. Se decide implementarlo en UDP debido a que no hace falta el uso de internet para poder iniciar la App.

3.4. tecnología ***HLS*** (*HTTP Live Streaming*).

HLS es una solución de streaming adaptativo basado en HTTP creado por Apple con el objetivo de realizar streaming adaptativo, es decir, la calidad multimedia se adapta en cada momento al ancho de banda que se utiliza en cada instante. La siguiente imagen se puede observar un escenario donde el servidor contiene todas las calidades disponibles del material multimedia para que en caso de que el ancho de banda varíe, el receptor pueda tener una reproducción continua pero con variaciones de calidad según el ancho de banda que esté disponible en ese instante.

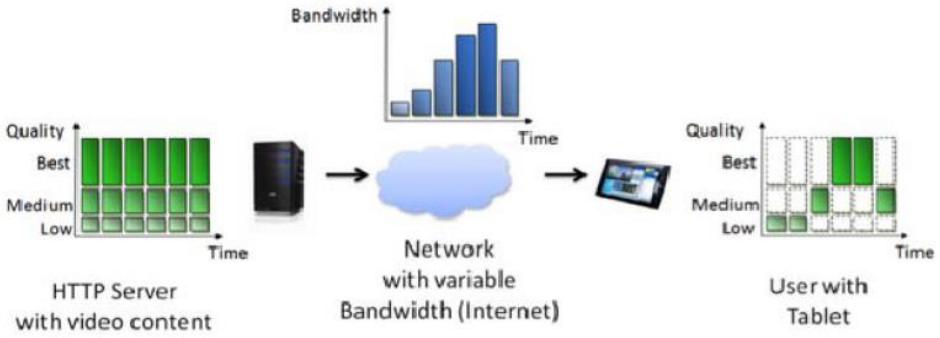

Imagen 11: Escenario del funcionamiento de la calidad de reproducción según el ancho de banda que se encuentre disponible

A medida que se reproduce el flujo, el cliente puede seleccionar entre un número de corrientes alternativas diferentes que contienen el mismo material codificado en una variedad de velocidades de datos, permitiendo que la sesión de transmisión se adapte a la velocidad de datos disponible. La imagen 11 hace referencia a la calidad mostrada según el ancho de banda que esté disponible en ese instante.

Al inicio de la sesión de *streaming*, HLS descarga una fichero con extensión .M3U8[1] extendida que contiene los metadatos para los varios sub-arroyos que están disponibles.

[1] es un formato de archivo que almacena listas de reproducción de medios. En un principio la posibilidad de crear y abrir los M3U solo era soportada por Winanmp, pero actualmente el formato es soportado por

Dado que sus peticiones sólo utilizan transacciones HTTP estándar, HTTP Live Streaming puede atravesar cualquier firewall o servidor proxy que permite a través del tráfico HTTP estándar, a diferencia de los protocolos basados en UDP como RTP (*Real-Time Protocol*). Esto también permite que el contenido se ofrezca desde servidores HTTP convencionales como origen y se distribuya a través de redes de entrega de contenido basadas en HTTP ampliamente disponibles

3.5. ***Servidor HTTP Apache***.

Es un servidor basado en HTTP y por tanto su acceso es universal para cualquier sistema operativo, entre ellos Linux, Windows, IOS, Android, es decir, es de código abierto y está desarrollado y mantenido por *Apache Software Foundation.* Su función permite el acceso a archivos por parte de los clientes solo con una dirección URL. En el caso de este trabajo, se utiliza para poder almacenar todo el contenido que empleará HLS para que posteriormente cualquier cliente tenga acceso a esos archivos para su reproducción.

3.6. Plataforma ***Gstreamer***.

En el capítulo anterior se han dado a conocer los elementos básicos que conforman Gstreamer así como su definición (2. Herramientas). Está compuesto de un núcleo y de una serie de plugins que proporcionan las diferentes funcionalidades, como pueden ser elementos de procesado de señal (p.ej., códecs, filtros…) o agentes de emisión y recepción para diferentes protocolos de red. Los plugins son librerías que se cargan dinámicamente en tiempo de ejecución. Gstreamer diferencia 4 categorías de plugins, divididas en paquetes independientes-

-*gst-plugins-base*: un conjunto de plugins esenciales con un mantenimiento actualizado, se trata de elementos básicos y elementales, como los que implementan la salida de imagen a la pantalla o de audio a los altavoces.

-*gst-plugins-good*: un conjunto de plugins de buena calidad bajo licencia LGPL. Un ejemplo son los elementos que comprueban la existencia de errores en el audio o el vídeo (audioparsers y videoparsers).

-*gst-plugins-ugly*: un conjunto de plugins de buena calidad, pero que puede plantear problemas de distribución por cuestiones de licencias o patentes. Un ejemplo es el elemento de codificación MP3 lamemp3enc.

-*gst-plugins-bad*: un conjunto de plugins que están en fase de desarrollo (alguno con soporte parcial de las funcionalidades buscadas) y necesitan más testeo. Un ejemplo es uno de los elementos utilizados en esta TFM, llamado mpegtsdemux, y que se encarga de de-multiplexar los distintos flujos MPEG2-TS.

Su uso está basado en la configuración de pipelines para desarrollar las aplicaciones multimedia. Cada pipeline puede contener una serie de elementos o componentes (definidos en los plugins) que, encadenados y de manera combinada,

múltiples reproductores, tales como: AIMP, foobar2000, iTunes, JuK, VLC, Windows Media Player, y XMMS, entre otros.

pueden realizar funcionalidades muy avanzadas. Es necesario entender el concepto de elementos para comprender el funcionamiento y filosofía de operación de GStreamer. Cada elemento es parte de un plugin, el cual proporciona algún tipo de funcionalidad (p.ej., reproducción, comunicación o edición de los contenidos multimedia), bien por sí mismo o bien en coordinación con otros elementos. Cada elemento puede tener uno o más puertos de entrada (sinks), por los que recibe los datos entrantes para ser procesados. De manera análoga, cada elemento puede tener una o más puerto (conocidos como Pads) de salida (sources), por los que proporciona los datos que ya ha procesado al siguiente elemento de la pipeline o de salida (p.ej., red, archivo, altavoces, pantalla...). Todos los elementos se conectan en cadena, por lo que entre dos elementos consecutivos existe una conexión. De esta forma, se consigue la implementación de aplicaciones, conectando todos los elementos necesarios para su ejecución (por ejemplo, partiendo de un archivo local, la lectura del contenido multimedia, su decodificación, procesado y reproducción). Además, cada elemento puede tener un número determinado de parámetros configurables. Por ejemplo, un elemento puede almacenar datos durante un tiempo determinado (por ejemplo, las colas de memoria) o puede modificar el contenido original (los elementos de codificación, que permiten diferentes parámetros). En concreto, los tipos de elementos disponibles en GStreamer son:

- Sources: generan, capturan o leen imagen y/o audio, o bien datos de la red.

- Formats: parsers, formaters, muxers, demuxers, metadatos, subtítulos, etc.

- Codecs: codificadores y decodificadores.

- Filters: convertidores, mezcladores, efectos, etc.

- Sinks: reproducen o guardan imagen y/o audio, o bien envían datos a través de la red.

Como ejemplo, en la imagen 12, muestra la pipeline necesaria para reproducir un archivo multimedia con formato OGG. En primer lugar, el elemento file-source es el encargado de leer los datos del archivo. A continuación, los datos se pasan al ogg-demuxer, que es el encargado de de-multiplexar audio y video, para enviarlo a sus respectivos decodificadores y, seguidamente, a las salidas de audio y vídeo. Asimismo, se puede observar que los elementos se unen a través de puertos.

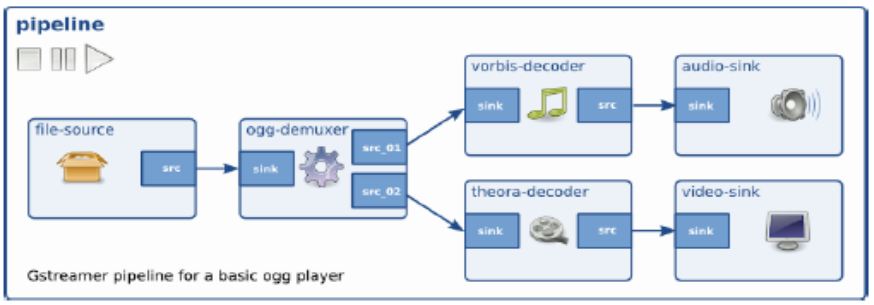

Imagen 12: Ejemplo de formato de la pipeline para poder reproducir un archivo almacenado

Capítulo 4. Solución desarrollada para el streaming IP de contenido recibido vía broadcast.

En este capítulo, se especifican los pasos seguidos para llegar a la construcción definitiva de las aplicaciones que componen la solución desarrollada en el TFG. Dicha solución consta de dos módulos bien diferenciados, uno destinado al servidor para ofrecer un servicio a los dispositivos cliente en el hogar; y otro distinto para que el usuario pueda interactuar con el servidor. En la figura siguiente se muestra el esquema final de la unión de los dos módulos. La parte de la izquierda representa el módulo del servidor y la de la derecha muestra los componentes del módulo cliente de un dispositivo basado en S.O. Android.

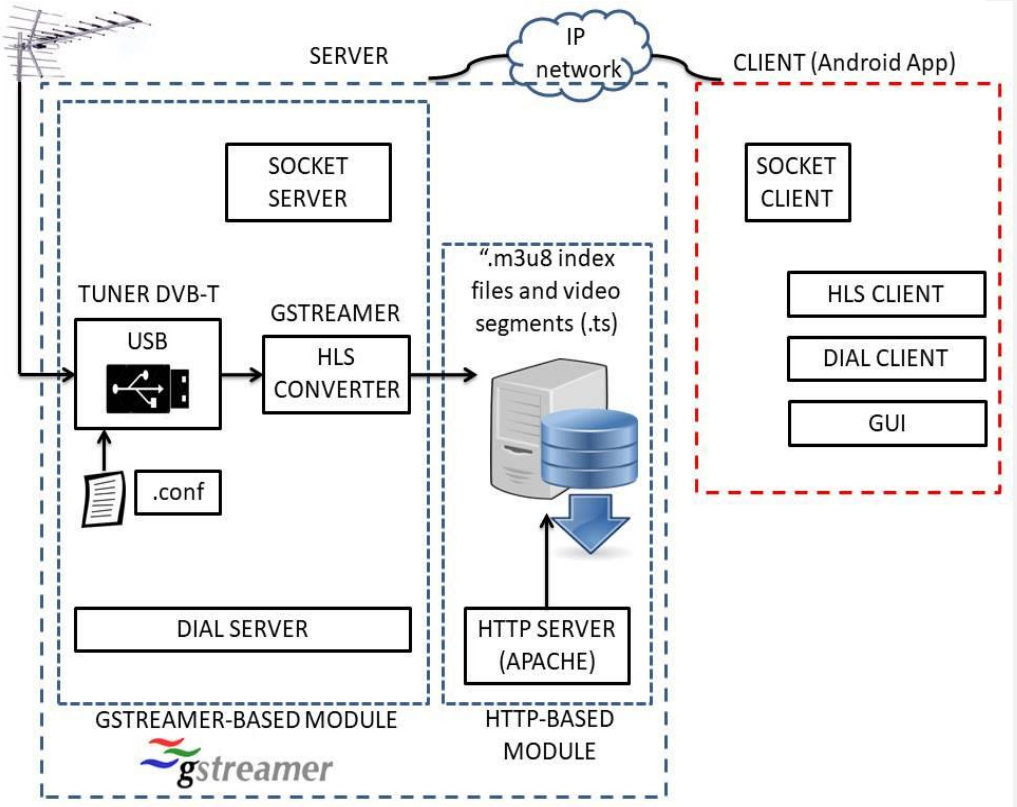

Imagen 13: Esquema de las dos Apps que conforman la solución planteada. A la parte de la izquierda se aprecia la App destinada a funciones de servidor. A la derecha contiene el apartado destinado al cliente.

Para una mejor descripción de la solución, en los siguientes apartados se describen el proceso individual que hace cada uno de los módulos que componen dicha solución final.

4.1. Módulo Servidor

Siguiendo con el desarrollo del TFG, el siguiente paso es describir el proceso de desarrollo e implementación de la App del servidor utilizando lenguaje de programación "*c"* en el entorno de Eclipse. Para ello, se describen los elementos que conforman todo el modulo correspondiente a la parte del servidor, relacionado con la parte izquierda de la imagen 13. Los elementos partícipes de la solución para el servidor son los siguientes:

- *DIAL Server* (Servidor DIAL).
- *Tuner DVB-T* (Demodulación DVB-T).
- *HLS Converter* (Convertidor a HLS).
- *Socket Server* (Servidor Socket).
- *HTTP Server* (Servidor Web).

A continuación se describen particularmente para poder entender mejor el proceso del funcionamiento total del módulo de la App del servidor,

- Servidor Dial. El primer paso en esta App es dar a conocer a todos los dispositivos que estén conectados en la misma red local donde se localiza el servidor, que éste puede ofrecer un servicio. Para ello el servidor nunca da comienzo hasta que un usuario (se entra en más materia en el apartado 4.2 Módulo cliente Android) lanza un mensaje de deseo de descubrir un servidor DIAL al iniciar la App móvil para que preste sus servicios, en este caso, el objetivo perseguido es la reproducción de contenido multimedia de tipo DVB-T.

A la hora de implementar este tipo de servidor, se ha pensado que es la mejor elección de cara al usuario para poder establecer un emparejamiento de forma transparente, es decir, sin la necesidad de vincular previamente los dispositivos. Este proceso tiene lugar de forma automática.

- Demodulación DVB-T. El siguiente paso a realizar por el modulo destinado al servidor, es la generación y análisis del archivo con extensión .conf (mencionado en el apartado 2.2 referente a herramientas de software). La escritura y lectura del archivo .conf, conlleva una serie de acciones desde el uso de la herramienta Dvb Tools hasta una creación de lista con información necesaria para completar la correcta emisión de los canales, es decir, se obtienen los elementos necesarios para la configuración de la pipeline de emisión. La información que presenta el archivo .conf, se presenta ordenada por canales y con el mismo número de elementos analizados por canal. Estos elementos que aparecen por canal van separados entre dos puntos, factor a favor a la hora de filtrar y clasificar la información para tratarla con más facilidad. La imagen 6 es un claro ejemplo de lo que se obtiene haciendo uso de esta herramienta. De toda la información que aporta, solo se va a realizar consumo de los siguientes

elementos. Para una mejor explicación, se expone la información extraída del multiplex de RTVE (Radio Televisión Española).

-Nombre del Programa (*La 1*): Es necesario este campo para que en la parte del cliente puede elegir según sus gustos, el canal que quiera visualizar. El servidor enviará la lista de nombres de canales y emisoras de radio que se han podido sintonizar. Se detalla este proceso más adelante.

-Frecuencia (770000000): Valor de la frecuencia en Hz de emisión del canal de TV o emisora de radio. Este valor es importante para la configuración de la pipeline encargada de transmitir todo el flujo de información de cara a la App para el cliente.

-PID de vídeo (101): Valor en decimal del correspondiente paquete que identifica el vídeo. En el caso de radio, este valor es 0.

-PID de audio (103): Valor en decimal del paquete identificativo del audio. El segundo PID de audio suele contener el idioma en VO (versión original), para obtenerlo basta con sumarle una unidad al valor del PID origen de audio.

-Número de programa (570): es el encargado de identificar cada programa visual dentro del mismo multiplex de emisión. Su valor esta expresado en formato decimal.

Una vez realizada la generación del archivo .conf y su análisis, el servidor estará en disposición de poder enviar toda la lista de nombres de programas de TV al usuario que ha exigido la prestación de sus servicios.

- Conversor HLS. Este apartado tiene lugar una vez ha sido generado, leído y analizado el archivo .conf anteriormente descrito. Con la información ya procesada, se da paso a la construcción del esqueleto que va a poseer la pipeline situada en el servidor encargada de realizar la retransmisión del canal seleccionado por el usuario desde su respectiva App.

El patrón que sigue la creación de la pipeline en este apartado referente a los canales de TV es el siguiente.

```
char *pipelineSD_parte1 = "dvbsrc adapter=0 inversion=AUTO modulation=AUTO trans-mode=AUTO bandwidth=8 frequency=";
//aqui el valor de freq
//... 626000000
char *pipelineSD_parte2 = " code-rate-lp=AUTO code-rate-hp=AUTO guard=AUTO hierarchy=AUTO ! queue ! tee name=tee0 ";
//aqui el nombre del tee
//... tee0
// a partir de la siguiente instruccion va sera cada rama, por lo que es lo que debe modificarse para cada canal
char *pipelineSD_parte3 = "tee0. ! queue ! tsparse ! tsdemux program-number=";
//aqui el program-number
//... 150
char *pipelineSD_parte4 = " name=";
//aqui el nombre del demuxer
//... demuxer
char *pipelineSD_parte5 = " ! queue ! mpegvideoparse ! mpegtsmux name=";//" ! queue !   mpeg2dec  !  queue ! avenc_mpeg
//aqui el nombre del muxer
//... muxer
char *pipelineSD_parte6 = " ! hlssink max-files=6 playlist-location=/var/www/html/HLStest/playlistSD";
//aqui se remata el nombre del fichero m3u8
//... antena3
char *pipelineSD_parte7 = ".m3u8 target-duration=4 location=/var/www/html/HLStest/segmentoSD";
//aqui se remata el nombre de los segmentos
//... antena3
char *pipelineSD_parte8 = "%05d.ts ";
//aqui se vuelve a meter el nombre del demuxer
//... demuxer
char *pipelineSD_parte9 = ". ! queue ! mpegaudioparse ! ";//".  ! queue !  mpegaudioparse ! mad ! audioconvert ! avenc_
//aqui se finaliza con el nombre del muxer
//... muxer
char *pipelineSD_parte10 = ". ";
```

Imagen 14. Captura de la preforma de la pipeline.

Con formato: Espacio Después: 0 pto

Por cada canal de TV, se utilizara el esquema anterior mostrado.

Mediante el intercambio de mensajes entre un módulo y otro, cuando el cliente recibe la lista de posibles canales de TV que puede visualizar, este realiza la elección de uno de ellos según sus gustos. Esta elección por parte del usuario da acontecimiento a la configuración final de la pipeline que se va a emitir, ya que con su elección, el nombre del canal de TV seleccionado, se puede filtrar todos los programas que pertenezcan al mismo rango de frecuencia de emisión comparándolo con el lista del archivo .conf

Por ejemplo, en el caso de que el cliente elija el canal de La 1 perteneciente al multiplex de RTVE, la configuración de la pipeline quedaría de la siguiente forma:

dvbsrc adapter=0 inversion=AUTO modulation=AUTO trans-mode=AUTO bandwidth=8 frequency=770000000 code-rate-lp=AUTO code-rate-hp=AUTO guard=AUTO hierarchy=AUTO ! queue ! tee name=tee0 tee0. ! queue ! tsparse ! tsdemux program-number=570 name=demuxer21 ! queue ! mpegvideoparse ! mpegtsmux name=muxer21 ! hlssink max-files=6 playlist-location=/var/www/html/HLStest/playlistSDLa1.m3u8 target-duration=4 location=/var/www/html/HLStest/segmentoSDLa1%05d.ts demuxer21.audio_00cb ! queue ! mpegaudioparse ! muxer21. demuxer21.audio_00cc ! queue ! mpegaudioparse ! muxer21.

Con formato: Espacio Después: 0 pto

Como se puede observar, el elemento *dvbsrc* contiene los siguientes subelementos:

- *adapter=0*: se indica que numero contiene de adaptador en el pc la sintonizadora USB.
- *inversion=AUTO*: se indica de forma automática la información de la inversión.
- *modulation=AUTO*: se establece la modulación de forma automática.

- *trans-mode=AUTO*: se establece el modo de transmisión de forma automática.
- *bandwidth=8*: indica el ancho de banda del multiplex en MHz.
- *frequency=626000000*: indica la frecuencia en Hz del multiplex.
- *code-rate-lp=AUTO* :se establece de forma automática la tasa de codificación de baja prioridad
- *code-rate-hp=AUTO*: se establece de forma automática la tasa de codificación de alta prioridad.
- *guard=AUTO*: se establece de forma automática el intervalo de guarda.
- *hierarchy=AUTO*: se establece de forma automática la información de jerarquía.

Queue : cola simple de datos.

Tsparse : analiza el flujo de transporte MPEG-2, permite comprobar que no hay errores.

Mpegvideoparse : identifica y comprueba que el flujo sea del tipo mpeg.

Tsdemux : decir que las subclases se ha puesto por investigación y para que pudiera funcionar. Se debe poner el program number correspondiente al flujo a visualizar, y además, se escoge que este elemento vaya compartiendo variables como el PTS con la aplicación desarrollada. Además se le da nombre a este elemento para poder tener una bifurcación en la pipeline, una rama para vídeo y la otra para audio.

Mpegtsmux : encargado de multiplexar el flujo multimedia de un programa en un TS.

Hlssink max-files=6 : indica que el número máximo de los archivos generados por *HLS.* En este caso son 6

Playlistlocation=/var/www/html/HLStest/playlistSDLa1.m3u8: indica en que raíz del directorio del servidor se encuentra el archivo con extensión .m3u8

target duration=4 : indica que cada segmento es de 4 segundos de duración

location=/var/www/html/HLStest/segmentoSDLa1%05d.ts:indica donde se almacenan los segmentos ts .El %05d indica que cada segmento tendrá 5 enteros en el nombre es decir que ira generando segmentos tipo antena300000.ts, antena300001.ts, antena300002.ts etc hasta conformar 5.

Esta pipeline sirve para el envío de un solo canal de TV. Para realizar el envío de cada canal se añade un elemento llamado *tee*(introducido ya en este ejemplo) donde su función es ramificar la pipeline, es decir, añade una rama por cada canal de TV del

mismo múltiplex En el caso de ser una emisora de radio, en su construcción se omite la creación de la parte del vídeo.

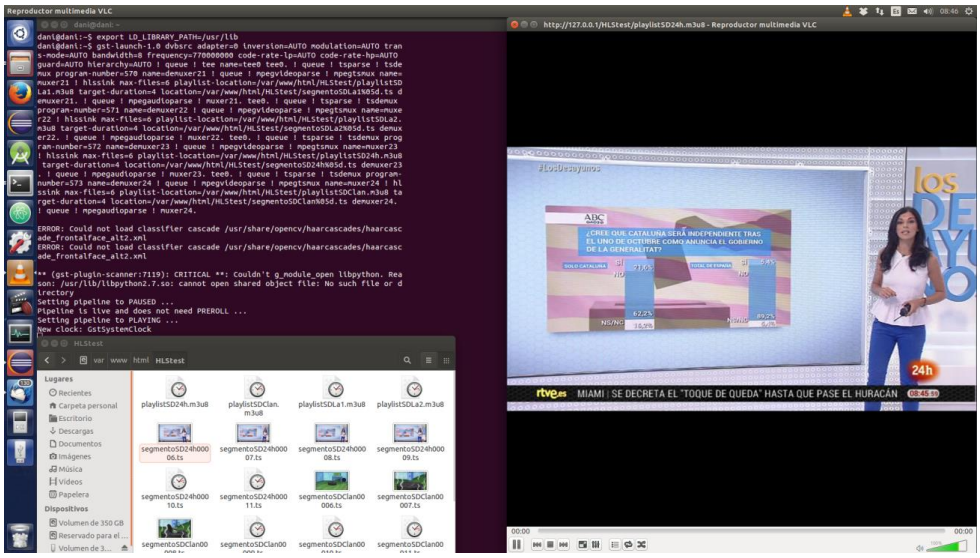

Imagen 15. Generación y reproducción de ficheros HLS con Gstreamer.

En el documento anexo se pueden observar más ejemplos de cómo realizar *streaming* mediante Gstreamer utilizando otras tecnologías, así como las pruebas previas al desarrollo del módulo del servidor para observar el comportamiento de la reproducción.

- Socket Server. Es el encargado de establecer el intercambio de mensajes entre el servidor y el cliente. Por cada mensaje se abre y se cierra el socket basado en UDP empleado para la comunicación entre extremos. Al ser UDP, no se garantiza el envío del mensaje, lo que si se asegura es que si se envía un mensaje llega seguro a su destino aunque no en el orden en el que se envía (protocolo no orientado a conexión)

Para esta comunicación se han definido tres tipos de mensajes que el cliente enviará al servidor: mensajes GET, POST y DELETE.

- Mensaje GET. El módulo cliente cuando ejecuta la App desarrollada realiza una petición de tipo GET y espera respuesta por parte del servidor. La respuesta del servidor es la lista de programas de TV disponibles para la

emisión, resultado de realizar (previamente) un barrido del espectro de radiofrecuencia mediante la tarjeta demoduladora de TDT instalada en el propio servidor.

Esta lista con todos los programas recibidos por antena sólo se enviará cuando no haya más clientes conectados al servidor. En el momento que exista un cliente ya conectado y recibiendo contenido de un programa, si, posteriormente, se conecta otro cliente al servidor, la lista de programas disponibles a enviar se verá reducida considerablemente, ya que sólo se enviarán a este cliente los programas que pertenezcan al mismo multiplex que el del programa de TV que haya seleccionado el primer usuario. Ello es debido a que la tarjeta sólo dispone de un sintonizador y sólo se puede seleccionar un canal UHF en cada momento, y, por consiguiente, acceder a un único múltiplex. Se emplea una variable que registra el número de clientes conectados en cada momento, para poder tener en cuenta cuantos usuarios hay visualizando contenido de TV.
Por ejemplo, si un primer cliente selecciona para visualizar entre todos los programas de TV disponibles, el programa de La 2, si posteriormente se conecta otro cliente distinto, la lista que recibe sólo contendría los programas La 1, La 2, Clan y 24h, ya que son los programas dentro de un mismo multiplex que corresponde a la compañía de RTVE emitido en el canal UHF correspondiente a una frecuencia central de 770 MHz.

Imagen 16. Envío mensaje GET desde el cliente (C) y el servidor (S).

En la imagen 16 se puede observar que el cliente envía el mensaje GET y la respuesta por parte del servidor es el envío de todas la lista de canales de TV en otro mensaje de tipo GET. Al recibir el mensaje el servidor, este incrementa una variable de tipo contador para poder controlar el número de usuarios que están haciendo uso de la App del servidor.

- Mensaje POST. Su función es poder notificar al servidor el programa seleccionado por el usuario de la aplicación cliente. El cliente cuando selecciona el programa deseado, una vez ya ha recibido respuesta del mensaje de tipo GET del servidor, envía el mensaje de tipo POST con la opción de programa de TV deseado. El servidor interpreta el mensaje y realiza la creación

de las pipelines correspondientes, introduciendo para cada canal de TV la información específica. Previamente a lo descrito en este mensaje, se comprueba el estado de una variable booleana denominada *transmisión* que se encarga de verificar que no esté ejecutándose una pipeline ya montada para dicho programa. Si la variable se encuentra modificada respecto su estado inicial(variable booleana), la pipeline ya estará montada, y, por tanto, no se realizará ninguna acción en el servidor. Por otro lado, si no ha cambiado su estado se realizará la creación de la pipeline con la configuración correspondiente.

Una vez creada la pipeline definitiva, se empezará a convertir el contenido del programa de TV recibido por antena a fragmentos de contenido según el estándar HLS (*HTTP Live Streaming*) [RFC 8216] a almacenar en el servidor HTTP. [6]
Una vez definido el formato del archivo que se va a reproducir, cabe mencionar que cuando se genera el contenido HLS se deja un cierto tiempo de espera que se comenta en el próximo apartado (*4.2 Módulo Cliente Android*). Cada vez que se inicie la aplicación por primera vez, todo contenido previamente almacenado, será borrado para evitar un mal funcionamiento y mostrar contenido que no ha sido seleccionado.

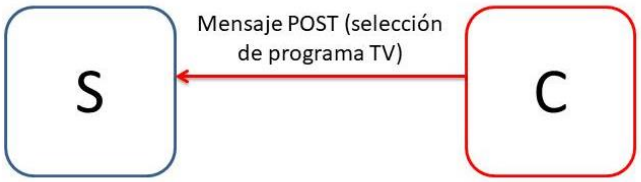

Imagen 17. Envío mensaje POST desde el cliente (C) y el servidor (S). El servidor no responde a este tipo de mensaje y crea la pipeline si todavía no esta en uso

Comentario [FBS1]: Pon el texto de las acciones del servidor en 1 línea. Tienes espacio de sobra

La imagen anterior muestra el envío del mensaje tipo POST (elección del canal por parte de la App para Android) indicando el canal a reproducir. El servidor por su parte no emite respuesta, sino que monta la pipeline para su retransmisión si esta, todavía no está montada. En el caso de que este en reproducción ya la pipeline, el servidor descartaría cualquier cambio en la configuración de la pipeline retransmitida.

- Mensaje DELETE. Este mensaje es enviado por la parte del cliente en dirección al servidor. Por parte del servidor, lo interpreta como una acción en la que el cliente muestra el deseo de abandonar la reproducción. La función que realiza el servidor es activar un restador que actúa sobre la variable de

usuarios conectados, es decir resta en una unidad cuando cada usuario quiere para la reproducción. Cabe destacar sobre Gstreamer, que una vez el conducto de la pipeline esta iniciado, no hay manera de modificarlo si no se termina la sesión por parte de todos los usuarios que están empleando el módulo del servidor. En la siguiente imagen se muestra gráficamente el mensaje de tipo DELETE.

Imagen 18. Envío mensaje DELETE desde el cliente (C) y el servidor (S). El servidor no responde a este tipo de mensajes. Si recibe un mensaje de este tipo, restará en una unidad el nº de usuarios

En este caso, el servidor no emitirá respuesta sobre el cliente, sino que sólo realiza la acción de restar el número de cliente que abandona la sesión.

- Servidor Web. Este último elemento perteneciente al módulo del servidor, es el encargado de almacenar toda la información HLS generada por Gstreamer, es decir, el material DVB-T transformado en HLS para la realización del *streaming*. El contenido del servidor son los archivos generados con extensión .m3u8 descritos anteriormente en el apartado de *Convertidor HLS*. Esta información almacenada, cada vez que se inicie por primera vez la App del Servidor, o cuando después de estar la pipeline activa todos los usuarios abandonen la sesión de uso del servidor, se borrará todos los ficheros generados que han se han ido almacenando en este servidor para poder evitar un mal funcionamiento por parte del módulo correspondiente al servidor.

El orden de aparición del contenido almacenado para reproducir es el que se comenta a continuación. En un primer instante, cuando el cliente ha seleccionado el canal, el servidor crea la pipeline dando lugar a la creación de un primer archivo de video (.ts) de una duración de 4 segundos por cada canal de TV perteneciente al mutiplex MPEG2 elegido (definidos según el esqueleto de configuración propia de la pipeline). Una vez se han generado esos ficheros, se da paso al inicio de la generación del archivo m3u8 que contiene información (metadatos) sobre los

segmentos de vídeo (.ts) generados. Este tipo de fichero no son generados hasta que se genere el segundo archivo ts correspondiente a cada canal.

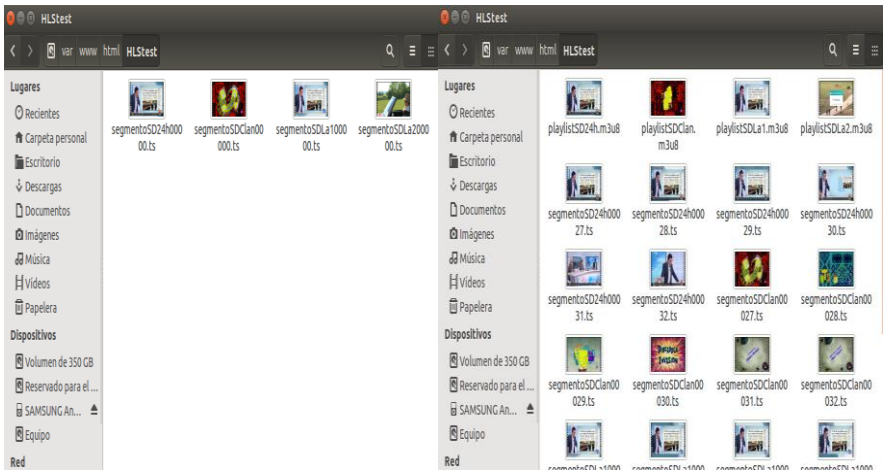

Imagen 19. A la izquierda ficheros iniciales ts, a la derecha ficheros ts con sus m3u8 generados.

4.2. Módulo Cliente Android.

Este apartado se dedica a explicar la arquitectura seguida para implementar una solución al módulo correspondiente al cliente. Los elementos que componen el cliente son el *cliente DIAL*, *cliente Socket*, *cliente HLS* y la *interfaz gráfica* empleada para la interaccionar con el servidor desde la parte del cliente. Estos conceptos se definen más detalladamente en las próximas líneas después de mencionar algunos motivos por los cuales se diseña la App del cliente para el sistema operativo Android.

Desarrollado el servidor, el siguiente paso en el desarrollo del trabajo es la implementación de una App para el uso en los dispositivos cliente. Con esta aplicación se pretende dar un servicio como el de poder visualizar la señal de tv a través de todo tu entorno de red, en este caso el uso doméstico. El usuario tendrá la posibilidad de ver contenido de tv online en cualquier dispositivo con conexión wifi y con un reproductor multimedia como VLC media player instalado.

Desde un principio, se decide implementar la App para el sistema operativo Android por los siguientes motivos:

- Se trata de un sistema operativo extendido a nivel mundial.
- Su código es abierto y con gran potencial para desarrollar aplicaciones.
- Contiene un gran número de desarrolladores, por lo que existe un gran número de páginas web de consulta fiables a las que poder acceder y utilizar su código.
- El desarrollo de aplicaciones no está cerrado a un sistema operativo determinado, como ocurre con iOS, que necesita un ordenador con OS X.
- Es un sistema operativo en constante actualización y mejora.
- Cuenta con una amplia documentación oficial para el desarrollador.
- Los terminales móviles y tablets que ejecutan este sistema operativo tienen un precio asequible.

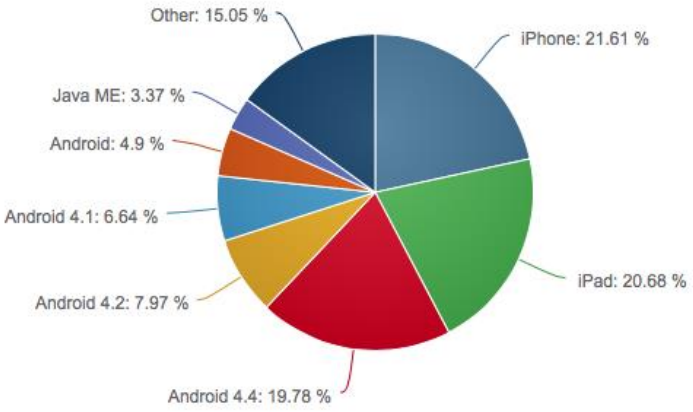

Imagen 210. Gráfico representativo del uso de de S.O.

Tal y como se ha indicado, se ha desarrollado un módulo cliente para dispositivos basados en sistema operativo Android. Para programar los componentes de la App de dicho módulo se ha empleado el IDE Android Studio, aunque también se podría haber utilizado el IDE Eclipse. Se decidió utilizar la plataforma de desarrollo Android Studio por los siguientes motivos:

- Se obtiene una renderización en tiempo real del diseño creado.
- Contiene una consola de desarrollador para la ayuda de traducción, consejos de optimización…
- Integración de Gradle en la construcción ya que es una herramienta capaz de automatizar dicha construcción de proyectos.
- Posibilidad de arreglos y compilación fluida.
- Plantillas preparadas para ser cargadas con varios diseños comunes de Android.

Nombradas algunas de las ventajas de usar Android Studio cabe destacar que posee un control de versiones donde permite realizar copias de seguridad almacenándolas en la nube. También puede restaurar a versiones anteriormente instaladas y es el sistema operativo más extendido entre dispositivos móviles, incluso por delante de IOS.

En conclusión, Android Studio es un entorno de desarrollo integrado para la plataforma Android. Fue anunciado el 16 de mayo de 2013 en la conferencia Google I/O, y reemplazó a Eclipse como el IDE oficial para el desarrollo de aplicaciones para Android.

Además de la instalación de la APK del módulo del cliente en el dispositivo móvil con sistema operativo Android (este proceso de instalación se puede consultar en la documentación anexa a este trabajo), es necesaria también la instalación de un reproductor lo bastante potente para poder reproducir el contenido audiovisual. Para este fin se ha pensado en el uso de *VLC media player,* ya que gracias a esta herramienta de licencia abierta, posee la gran capacidad de reproducir en su totalidad un gran número de diferentes tipos de archivos.

Después de haber expuesto algunos motivos por el que se implementa esta App para su uso en Android, se pasa a describir su funcionamiento. Para ello, como se ha comentado al principio de este capítulo, se pasa a remarcar la función que comete cada elemento

- Cliente DIAL. Este elemento tiene lugar de inicio en el momento que el usuario inicia la App propia del cliente. De esta forma, se da por entendido que el cliente quiere obtener los servicios que pueda ofrecer este servidor. Cuando se arranca la App para el cliente, de manera automática y totalmente transparente de cara al usuario, quedan vinculados cliente y servidor, si no existe ningún problema en la conexión. Esta vinculación resulta ser más cómoda para el consumidor ya que solo tiene que dar inicio a la App del cliente para establecer comunicación con el servidor sin tener que enlazar los dos dispositivos (cliente-servidor) de forma manual. Esto se consigue gracias a que el usuario al abrir la App manda un mensaje de descubrimiento de servidor DIAL. El mensaje se envía de forma multicast, para poder asegurarse que

dentro del mismo entorno de red local se pueda encontrar el servidor. En la siguiente imagen se puede observar el uso de DIAL en este trabajo.

Imagen 21. Protocolo DIAL

Al realizar uso de DIAL en módulo del cliente se puede identificar los procesos de su funcionamiento a través del log. El envío del mensaje por parte del cliente aporta la siguiente información:

```
09-11 10:24:06.272 8617-8669/com.tfg.carlos.castviewer D/THREAD: Lanzamos busqueda M-SEARCH
09-11 10:24:06.272 8617-8669/com.tfg.carlos.castviewer D/sendMSearchMessage: Dentro de sendMSearchMessage
09-11 10:24:06.277 8617-8669/com.tfg.carlos.castviewer D/SSDPSocket: SSDPSocket creado
09-11 10:24:06.277 8617-8669/com.tfg.carlos.castviewer D/SSDPSearchMsg: M-SEARCH * HTTP/1.1
        HOST: 239.255.255.250:1900
        MAN: "com.tfg.carlos.ssdp:discover"
        MX: 5
        ST: urn:dial-multiscreen-org:service:dial:1
        USER-AGENT: Android Dial Client for discovery
09-11 10:24:06.277 8617-8669/com.tfg.carlos.castviewer D/SSDPSocket: Mensaje recien enviado
09-11 10:24:06.277 8617-8669/com.tfg.carlos.castviewer D/sendMSearchMessage: Enviados paquetes de busqueda
09-11 10:24:06.277 8617-8669/com.tfg.carlos.castviewer D/SSDPSocket: Esperando paquete respuesta...
```

Como se ha comentado anteriormente, la respuesta a este mensaje es la dirección IP donde se localiza el servidor. La siguiente información es la vista del log por parte del cliente cuando recibe la respuesta.

```
09-11 10:24:06.357 8617-8617/com.tfg.carlos.castviewer D/IP Servidor: 192.168.0.108
09-11 10:24:06.357 8617-8669/com.tfg.carlos.castviewer D/sendMSearchMessage: +IP_SERVIDOR
09-11 10:24:06.357 8617-8617/com.tfg.carlos.castviewer D/*: ip DEL SERVIDOR: 192.168.0.108
```

- Cliente Socket. Este apartado corresponde a la construcción de un canal para la comunicación entre cliente y servidor para controlar el material multimedia que se

envía para dar solución a la petición del usuario. Como se ha mencionado en el capítulo 3, por cada mensaje que se ha enviado, previamente se ha abierto y cerrado una conexión UDP (socket) para poder cambiar información y elecciones de acciones entre los dos módulos. Esto se debe al uso de los mensajes que se han definido anteriormente.

En todo momento, la parte del cliente es la encargada de emitir siempre todos los mensajes. Su función es de control de la App. En primer lugar emite el mensaje de tipo GET, el cual se ha enviado de forma automática después de haberse iniciado la App desde la parte del cliente. Destacar que previamente al envío de mensajes tiene que haber pasado por el proceso de DIAL con total éxito. Si no, no se da suceso al envío de mensajes puesto que no se ha establecido una jerarquía de funciones entre los elementos extremos que son el cliente y el servidor. A continuación, se puede observar la petición lanzada desde Android Studio al servidor y la respuesta obtenida:

```
09-11 10:24:06.357 8617-8670/com.tfg.carlos.castviewer D/THREAD: Lanzamos peticion
Lista_canales

09-11 10:24:06.362 8617-8670/com.tfg.carlos.castviewer D/GET response:
tdp:tdpHD:RadioClasicaHQ:Radio3HQ:RadioExteriorRNE:CanalIngenieria:9KissTV:KissFM:10:an
tena3HD:antena3:laSextaHD:laSexta:neox:nova:Telecinco:Cuatro:FDF:Divinity:TelecincoHD:Cua
troHD:La1:La2:24h:Clan:La1HD.:RadioNacional:Radio5
09-11 10:24:06.362 8617-8617/com.tfg.carlos.castviewer D/Numero de canales disponibles:: 28
```

El segundo mensaje, el de tipo POST, aparece en el instante en el que el servidor ha contestado al primer mensaje GET que se ha emitido desde el usuario. Como respuesta por parte del servidor, en el cliente se ha obtenido la lista entera de canales de TV que ha podido sintonizar. En este instante, el usuario podrá elegir visualizar un programa según sus deseos o gustos. Fruto de su elección, se da paso al envío del mensaje de tipo POST.

```
09-11 10:24:10.277 8617-8617/com.tfg.carlos.castviewer D/onClick: Seleccionado canal La1
09-11 10:24:10.277 8617-8743/com.tfg.carlos.castviewer D/THREAD: Lanzamos peticion
reproduccion
09-11 10:24:10.277 8617-8743/com.tfg.carlos.castviewer D/Canal seleccionado: POST La1
```

Una vez reproducido el canal de TV en el cliente, se puede dar el caso que otro cliente quiera usar la App. En este caso, el nuevo cliente realiza los mismos pasos citados anteriormente, pero la respuesta del mensaje GET que recibe el cliente no es la lista entera de canales de TV sino que obtiene una lista filtrada ya que un usuario principal ha anticipado su elección provocando que solo se pueda escoger un canal de TV que contenga su mismo rango de frecuencia central de emisión que el elegido por el primer usuario. El cliente no obtendrá repuesta por parte del servidor a la petición

POST. El siguiente suceso que procede en la parte del cliente es la reproducción del canal detallado en el siguiente apartado, Cliente HLS.

Por otro lado, si el usuario abandona la sesión de reproducción o se ha producido un error en la reproducción, el cliente invoca el mensaje de tipo DELETE. El servidor trata este mensaje como barrera de control para obtener información de los usuarios que hacen uso de la App. Los mensajes generados por la parte del cliente se puede observar en los siguientes comandos extraídos del *Log* de Android.

```
09-11 10:24:17.142 8617-8854/com.tfg.carlos.castviewer D/THREAD: Lanzamos peticion de desconexion
09-11 10:24:17.142 8617-8854/com.tfg.carlos.castviewer D/Orden lanzada: DELETE
```

- Cliente HLS. Esta parte del módulo del cliente es la correspondiente a la parte final de la función de la App para el usuario. Sucede cuando por parte del servidor se ha empezado a enviar el contenido multimedia solicitado por parte del cliente.
Una vez escogido el programa de TV hay un tiempo de espera asignado, lo suficiente extenso y no molesto para dar tiempo a la creación de los archivos con extensión .m3u8 mencionados en el apartado anterior relacionado con el módulo del servidor. Seguidamente, salta un aviso en la parte del cliente donde se da la opción para elegir el reproductor que quiere emplearse para la reproducción. Aquí es donde entra en juego el uso de *VLC*. La reproducción del contenido multimedia se realiza a través de esta herramienta.

Imagen 22. Captura de pantalla desde el dispositivo móvil en el instante de elección de reproductor.

La reproducción es condicionada por el mensaje tipo POST enviado por parte del cliente, ya que la información de tipo POST que se recibe por parte del servidor es la dirección URL donde quedan almacenados los archivos con extensión .m3u8.

Ejemplo de url: http://IP_SERVIDOR/HLStest/NOMBRE_CANAL.m3u8

Como se puede observar, los campos escritos en mayúscula serán la información dinámica que cambiara según la elección que se haya escogido. El resto de directorio raíz permanecerá estático para así poder completar la ruta específica para una buena comunicación. Resumiendo, sólo cambiaran los elementos IP_SERVIDOR obtenido por DIAL, y NOMBRE_CANAL que se obtiene de la elección por parte del cliente.
En la imagen siguiente se expone una captura de pantalla realizada desde el terminal móvil donde se ha ejecutado el módulo diseñado para el cliente.

- *GUI* (Interfaz Gráfica de Usuario). En este apartado se presentan capturas de pantalla realizadas por el terminal móvil empleado para las pruebas en el laboratorio de la Universidad. Se trata del terminal Samsung S5 tratado en el capítulo 2 dedicado a las herramientas utilizadas para la realización de este TFG. En la siguiente imagen se observa el resultado que se obtiene al iniciar la App del módulo del cliente.

Imagen 23. Captura de pantalla desde el dispositivo móvil al inicio de la App del cliente.

Con formato: Izquierda, Sangría: Primera línea: 0 cm

Una vez elegido el programa de TV, no se podrá realizar la visualización de cualquier otro programa fuera del multiplex ya empleado

Para una mejor descripción del funcionamiento, se da por supuesto que un primer usuario hace uso de la App y elige el canal de La 1 para ser visualizado. Después de recibir el contenido audiovisual, si otro usuario empieza a utilizar la App, sin que el otro usuario abandone la sesión, la lista que recibe de canales disponibles serán *La 1, La 2, Clan* y *24h* (incluidos en el múltiplex recibido por misma frecuencia de emisión o canal UHF). En la siguiente imagen se observa el resultado final del ejemplo descrito anteriormente.

Imagen 24. . Captura de pantalla desde otro dispositivo móvil estando iniciada ya la App desde otro terminal.

Finalmente, después de realizar la elección del canal, el usuario recibe en su propio terminal la elección demandada. Su visualización se realiza a través de la app mediante una llamada al reproductor VLC a pantalla completa, presentado la información de la manera que aparece en la siguiente imagen

Imagen 25. Captura de pantalla desde el dispositivo móvil en el instante de reproducción del flujo multimedia seleccionado

Si cabe la posibilidad, si el programa que se encuentra decodificado está doblado, es decir, no está en su formato de origen, desde el menú de VLC deja elegir la pista de audio que esté disponible. He aquí una imagen que muestra esta opción.

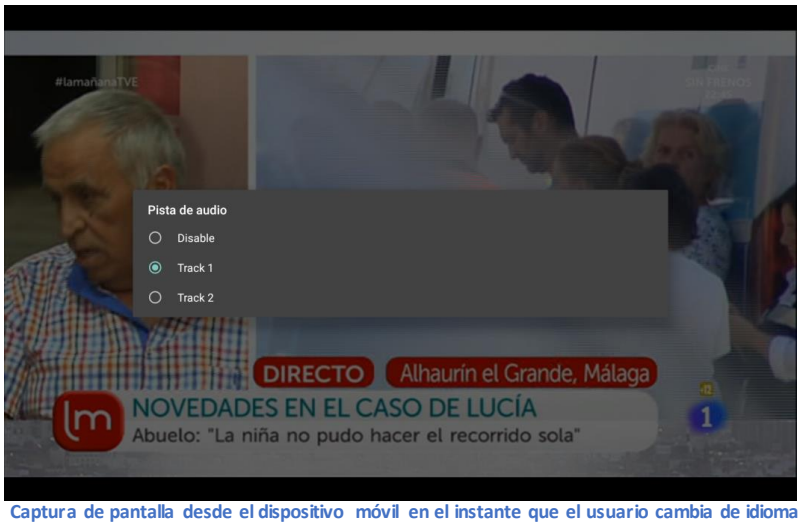

Imagen 26. Captura de pantalla desde el dispositivo móvil en el instante que el usuario cambia de idioma en la reproducción.

Capítulo 5. Conclusiones y líneas futuras.

El uso de la propuesta desarrollada en el TFG da una solución al problema de poder estar en una misma habitación con toma de antena viendo distintos programas de TV (incluso sin un TV disponible en la misma). Con el uso de cualquier dispositivo con conexión wifi, como puedan ser Smartphones, Tablets o incluso ordenadores, se puede obtener la señal de TDT en su dispositivo. En una misma estancia, varias personas pueden disfrutar de esa señal sin necesidad de marcharse a otro lugar de la vivienda, donde pueda existir otra televisión. Aparte, también presenta solución al problema de la visualización de contenido para adultos cuando haya usuarios menores, ya que no será excusa para el menor el tener que ir a otra instancia del domicilio para ver otro contenido de TV y podrá estar junto con sus tutores responsables para poder realizar su respectiva supervisión.

En la actualidad, cada canal de TV suele tener su propia web o App de emisión del canal pero la gran afluencia de usuarios ocasiona paradas en la reproducción, mala calidad o en casos de saturaciones del servidor, no llegar ni si quiera a funcionar. La App desarrollada puede ofrecer, aparte del servicio de reproducción de la señal TDT, resolver el problema que padecen esas App o web de las emisoras de TV puesto que tienen que estar invirtiendo en tecnología constantemente para poder dar ese servicio. La App hace que cada domicilio tenga su propio servidor con el que los problemas de congestión de servidores en Internet será mucho menos elevado que con los servidores que utilizan las anteriores aplicaciones mencionadas.

Por otro lado, destacar que con el uso de esta solución también se pretende ayudar a la sociedad en los siguientes aspectos:

- Integración social de la gente para la adaptación a otras lenguas extranjeras de comunicación verbal, ya que se puede dar el caso de tener serias deficiencias al tratar de usar un nuevo lenguaje comunicativo y tener necesidades previas a practicar la escucha de vocabulario en dicho idioma. Se puede observar que se da solución, como se acaba de explicar anteriormente, al problema de sociabilidad que padecen los jóvenes del presente (permanecen cerrados en sus respectivas habitaciones).

-Mejoras sensoriales para personas que presenten alguna deficiencia auditiva. Podrían escuchar, mediante auriculares conectados, ya sea a con una Tablet o Smartphone, el programa deseado sin tener que aumentar el volumen del televisor y así evitar molestar a las personas que se sitúen a su alrededor.

-Mejoras para personas con deficiencias visuales. Esta App permite visualizar el contenido en dispositivos secundarios (*secondary screens*), por lo que el usuario podrá tener en sus manos el material que visualiza y, por tanto, tendrá todo más a su alcance. Esta situación se puede dar tanto en un ámbito doméstico, como sería el caso de estar en casa y no llegar a ver bien la TV por la distancia a la que se encuentra, o incluso en ámbitos de ocio, donde su situación influye o el sonido ambiente es muy alto para poder escuchar con detenimiento lo que se está emitiendo.

Una vez comentadas situaciones problemáticas solventadas por el uso de la solución desarrollada, cabe mencionar el estado de los objetivos planteados al

principio de la realización de este trabajo. Para ello, se vuelven a exponer y en su caso verificar su cumplimiento.

1. *Creación de una App que permita emitir flujo multimedia de tipo DVB-T y que haga disponible su contenido vertiendo a través de la red local. Esta es la parte dedicada al servidor.*

   - Este objetivo se ha cumplido parcialmente, puesto que la App sólo se ha desarrollado para soportar los canales que están en SD (*Standard Definition*). El PC con el módulo servidor instalado, usado para realizar las pruebas es de bajo coste (miniPC), y, por tanto, su rendimiento es medio-bajo, además de ocasionar en muchas ocasiones sobrecalentamiento de terminal y apagados forzados cuando se intentaba trabajar con contenido HD.

2. *Creación de una App que permita seleccionar y recibir contenido multimedia a través de la información que ofrece la parte del servidor*

   - Dicho objetivo se ha cumplido totalmente. Cabe mencionar el gran papel que ofrece VLC gracias a los recursos que abastece. Sin esta herramienta, se tendría que haber implementado un reproductor propio en la parte del módulo cliente para Android, lo cual hubiera aumentado la complejidad de este trabajo.

Aparte de estos objetivos, destacar que el estudio de tecnologías empleadas para el desarrollo del funcionamiento de las Apps ha supuesto inversión de mucho tiempo en investigar, analizar y poner en conocimiento todo recogido en este trabajo. De todas las tecnologías que se ha hecho uso, solo dos (*MPEG2 y HLS*) han sido aprendidas en el Grado. De DIAL y sockets UDP se han obtenido los pasos a seguir gracias a los consejos y tutorías ofrecidas por los tutores asignados. A demás de aportar información de provecho, han marcado las pautas a seguir para acometer la realización de un App multimedia desde 0. Gracias a este paso, se puede considerar que se han obtenido los suficientes conocimientos para poder desempeñar funciones de desarrollador de Apps multimedia. La plataforma Gstreamer ha sido la más estudiada, de la cual se presente en un anexo toda la información recogida.

Sobre las tecnologías empleadas, se puede afirmar que se ha entendido su aplicación para obtener un buen trabajo para su posterior uso, por ejemplo, el enlazado que tiene lugar en los dispositivos secundarios, la generación de contenido accesible, la comunicación empleada tanto por cliente como servidor, arquitectura de cada protocolo como DIAL, HTTP, apertura de sockets …
También se ha aprendido la instalación y manejo de otro S.O. diferente a Windows (del que era usuario habitual), como es Ubuntu (Linux), además del manejo de todo el software mencionado en el capítulo 2.

Por otra parte, al principio, se experimentó la creación de la parte de desarrollo de la App para el usuario, la compilación de Gstreamer, pero visto su gran restricción de uso en programación para Android se declinó esta opción. Se han realizado pruebas de emisión utilizando el protocolo RTP, obteniendo malos resultados de emisión. Pese

a esta situación, la solución que se eligió fue la de utilizar de HLS frente RTP. En la documentación anexa se pueden encontrar pipelines relacionadas con el manejo multimedia usando este protocolo. Seguidamente, se muestran algunas razones por las que se ha decidido emplear HLS.

- El uso de Gstreamer mediante HLS es mucho más eficiente que empleando RTP.

- El servidor apache que hace uso HLS es de licencia abierta y gestiona mejor los recursos frente a cualquier servidor que se pueda implementar a través del framework de Gstreamer.

Para terminar, sólo queda por mostrar las posibles implementaciones al proyecto en un futuro si persigue el desarrollo de la App.

- Una de las aportaciones que generaría atracción por el uso de esta App sería la opción de introducir la pista de los subtítulos del canal que se encuentre reproduciendo. Dicha función ofrecería más servicios orientados al aprendizaje de idiomás.
- Otra línea futura que se deja por resolver, es la reproducción de contenido sintonizado en HD (*High Defintion*).
- La introducción de otro sintonizador aumentaría la cantidad de canales que se pueden visualizar en varios dispositivos en un mismo instante, al poder retransmitir el contenido de los programas de dos multiplex simultáneoamente.

Para concluir, no se detallan en este trabajo los resultados obtenidos del uso de esta App puesto que el mes de agosto ha permanecido el laboratorio cerrado y no ha dado tiempo de redactarlos para su posterior análisis. Estos serán expuestos el día de la defensa del trabajo frente el tribunal.

En conclusión, sólo queda destacar que aparte de investigar sobre tecnología de retrasmisión de señal televisiva, uso de Gstreamer, instalaciones de software, se han tomado siempre las mejores decisiones tanto de herramientas para dar lugar a un desarrollo e implementación de este trabajo, como ofrecer un servicio minimizando costes y precios para así poder dar una serie de prestaciones.

Referencias Bibliográficas.

- Jhon Watkinson, “MPEG-2, Butterworth-Heindelberg”,Focal press, 244 pág, ISBN 0 240 51510 2 . (año 1999)
- Jan Van der Meer, “Fundamentals and Evolution of MPEG-2 Systems: Paving the MPEG Road Wiley editorial, 464 pág, ISBN 978-0-470-97433-9(May 2014).
- Recommendation ITU-T H.222, “Tecnología de la información - Codificación genérica de imágenes en movimiento e información de audio asociada: Sistemas”, 2014.
- ETSI TS 102 823 - DVB (2010), European Telecommunications Standards Institute, ETSI, “Specification for the carriage of synchronized auxiliary data in DVB transport streams”
- https://as.com/betech/2016/09/06/portada/1473121939_146516.html
- http://www.panoramaaudiovisual.com/2015/10/22/el-53-de-espanoles-ven-television-y-video-en-streaming-a-diario/
- http://wikitel.info/wiki/MPEG_2
- https://ca.wikipedia.org/wiki/Program_Stream
- https://es.wikipedia.org/wiki/Transport_Stream
- https://oscar1110.files.wordpress.com/2010/07/transportpacket.jpg
- https://en.wikipedia.org/wiki/HTTP_Live_Streaming
- https://gstreamer.freedesktop.org/
- http://docs.gstreamer.com/display/GstSDK/Tutorials
- http://www.ubuntu.com/download/alternative-downloads
- https://eclipse.org/
- https://developer.android.com/studio/index.html
- https://sourceforge.net/projects/dvbinspector/files/
- https://sourceforge.net/projects/dvbtools/?source=directory
- https://es.wikipedia.org/wiki/Digital_Video_Broadcasting
- https://es.wikipedia.org/wiki/MPEG-2
- https://www.dvb.org/standards
- https://www.netway.com.es/descargas/sintonizador-tdt-netway-usb2-0
- http://docs.gstreamer.com/display/GstSDK/gst-inspect
- Material docente de la asignatura de Flujos de Datos Multimedia, impartida por el profesor Fernando Boronat, en curso 2016/17.
- https://developer.android.com/reference/java/net/MulticastSocket.html
- https://en.wikipedia.org/wiki/MPEG-1
- https://gstreamer.freedesktop.org/documentation/rtp.html
- https://blog.uchceu.es/informatica/ranking-de-sistemas-operativos-mas-usados-para-2015/