Desarrollo de un sistema de sincronización de subtítulos para subtitulación de programas de TV en directo

Autor: Alejandro Lorenzo Prada
Tutor: Mercedes de Castro Álvarez

DEPARTAMENTO DE INGENIERÍA TELEMÁTICA

PROYECTO FIN DE CARRERA
UNIVERSIDAD CARLOS III DE MADRID
INGENIERÍA DE TELECOMUNICACIÓN

Leganés, 11 de julio de 2011

Título: Desarrollo de un sistema de sincronización de subtítulos para subtitulación de programas de TV en directo.

Autor: Alejandro Lorenzo Prada.

Tutor: Mercedes de Castro Álvarez.

EL TRIBUNAL

Presidenta: María Calderón Pastor

Vocal: Belen Ruiz Mezcua

Secretaria: Ana Mª Iglesias Maqueda

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 11 de julio de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Dedicado a mis padres, Mayte y Alejandro

Agradecimientos

En estas líneas me gustaría agradecer todo el apoyo recibido en los últimos años durante el largo camino recorrido para llegar a esta importante meta de vida. Resulta complicado acordarse de todas las personas que en algún momento me han alentado a seguir adelante en los momentos difíciles, pero que no le quepa duda a nadie que estaré eternamente agradecido a todos y cada uno de ellos.

Durante el último año, he conocido a un gran número de personas en el CESyA a las que quiero dar las gracias por ayudarme y confiar en mí para sacar adelante este proyecto: a Mercedes, por su confianza y ánimo en aquellos momentos de "oscuridad", a Juanfran, que ha sido un auténtico apoyo y fue el perfecto compañero contando unos y ceros, y a Juanma y Alberto, que han sido los mejores compañeros de laboratorio que podía haber tenido, facilitándome la adaptación al entorno del CESyA desde el primer día y constituyendo un verdadero deshago en los momentos de frustración en que las cosas no salen como uno desea. Para el resto del personal de CESyA, gracias por ser como sois. Ha sido todo un privilegio poder llevar a cabo el proyecto junto a vosotros.

No me quiero olvidar de mis compañeros de la Universidad, con los que he compartido, no sólo largas horas de clases, prácticas, exámenes, sino también grandes momentos de mi vida. Me gustaría dedicaros unas palabras a cada uno de vosotros, pero la lista es extensa y no puedo extenderme en exceso. Destacaré a mi eterno compañero de prácticas. Marcos, mil gracias por ser como eres, por tus soluciones mágicas en las prácticas, incluso aquellas que no funcionaban. Si con alguien podía ser llevadero acabar una práctica a las 7 de la mañana tenía que ser contigo.

A mis amigos de toda la vida no hace falta que les diga nada, porque saben de sobra lo que representan para mí. Llevamos casi toda la vida juntos, algunos incluso desde la guardería. A vosotros también os debo el haber llegado hasta aquí.

A mi familia, por preocuparos siempre por mis evoluciones y darme vuestro apoyo y ánimo, en especial a mi abuela, que siempre ha estado ahí haciéndome sentir orgulloso de ser su nieto. Muchas gracias a mis tíos, padrinos y primos, y sobretodo,

a uno al que nunca olvidaré y que siempre fue mi héroe desde pequeño.

A mi novia, que me ha empujado en los momentos en que más lo necesitaba durante los dos últimos años de carrera, acompañándome incluso en mis jornadas nocturnas de estudio. Muchas de ellas, sé que estabas ahí simplemente por alentarme y hacerme más llevaderos esos momentos, por lo que nunca podré devolverte todo lo que me has dado. Tú me diste la energía que necesitaba para continuar cuando no creía ser capaz de lograrlo.

Finalmente, quiero agradecer a mis padres ser lo que soy. Sin su educación y buen hacer y sin su apoyo, no sería la persona que hoy soy, ni habría conseguido alcanzar muchas de las metas que me he propuesto a lo largo de mi vida. Gracias por guiarme en mi vida y por vuestro apoyo incondicional.

Resumen

El subtitulado de programas de televisión en directo constituye el principal reto tecnológico actual para la accesibilidad de programas que se emiten en tiempo real en las cadenas de televisión. En la mayoría de estos programas no existe la posibilidad de anticipar la transcripción a texto del audio del programa para su posterior paso a subtítulos y difusión en la señal de televisión.

Aunque existen diferentes apoyos tecnológicos y metodologías asociadas para la conversión de audio a texto en tiempo real, la principal alternativa de cara al subtitulado masivo de programas televisivos en directo es el uso de herramientas basadas en reconocimiento automático del habla (ASR). Las herramientas de ASR, aplicadas a la señal de audio del programa televisivo, permiten generar una transcripción del discurso, pero siempre con un retardo respecto al audio original. Existe sin embargo la posibilidad de generar una versión del canal de TV en la que el vídeo, el audio y los subtítulos se hagan llegar al usuario de manera sincronizada.

En este proyecto se exploran las posibilidades que ofrece DVB y el subtitulado DVB para la obtención de un flujo de subtítulos DVB-SUB sincronizado con los flujos de vídeo y audio, y su posterior distribución en redes IPTV. Dentro del subtitulado DVB se pondrán a disposición del usuario dos alternativas de codificación: la basada en píxeles (imágenes) y la basada en cadenas de caracteres.

Los resultados del proyecto permiten concluir que la sincronización de subtítulos en tiempo real es viable, y que la obtención de un flujo de televisión con subtítulos sincronizados puede ser implantado en los programas de televisión en directo. Por otro lado, a la vista de de los resultados obtenidos, en futuros sistemas de subtitulado automático sincronizado se recomienda el uso de la codificación basada en píxeles, dado que es la que se encuentra ampliamente extendida entre los fabricantes de receptores TDT y la que además, ofrece mejores prestaciones al público al que va dirigido.

Palabras clave: ASR, DVB, DVB-SUB, IPTV, Automático, Sincronizado.

Abstract

Nowadays, subtitling in live television represents the main technological challenge for the accessibility of real time TV programs. In the great majority of these programs there is no possibility of bringing forward the text to audio transcription of the program so that it could be turned into subtitles and finally be broadcast.

Although there exist different technological supports and its associated methodologies for audio to text conversion in real time, the main alternative in mass live television programs subtitling is the use of tools based on the automatic speech recognition (ASR). ASR tools, applied to the audio signal of the TV program, allow generating a transcription of the speech, but always with a delay respect to the original audio. However, there exists the possibility of generating a program version in which the video, the audio and the subtitles are synchronized.

In this project we will investigate the possibilities offered by DVB and the DVB subtitling system in order to obtain a DVB-SUB subtitle stream synchronized with the video and audio streams, and its subsequent distribution in IPTV networks. Within DVB subtitling, the project will offer to the user two different coding alternatives: coding of pixels and coding based on strings of characters.

The achieved results allow us to conclude that synchronization in real time subtitling is feasible and it could be applied to live television programs in order to achieve a television stream with synchronized subtitles. On the other hand, considering the achieved results, in future automatic synchronized subtitling systems it is recommended to use the coding based on pixels, since it is widely spread out among the DVB receiver manufacturers and, what is more, because it offers better performance to the audience.

Keywords: ASR, DVB, DVB-SUB, IPTV, Automatic, Synchronized.

Índice general

Índice de figuras

Índice de tablas

Parte I

Introducción

Capítulo 1

Introducción

1.1. Proyecto

Desde 1990 aproximadamente en España disfrutamos de los subtítulos en la televisión. Algo tan básico hace que nos sintamos más integrados debido a que dentro de lo que llamamos sociedad de la información estamos todos: personas mayores, niños, personas del extranjero en busca de aprender el idioma, personas con discapacidad auditiva o personas en las que dicha discapacidad no es muy pronunciada pero sienten la necesidad de algo que les ayude.

Según este perfil, hoy en día en España existen varios millones de personas discapacitadas, número que aumentará en el futuro como consecuencia del envejecimiento de la población. Por ello, la prestación de servicios de accesibilidad ha de contemplarse no sólo como una cuestión de igualdad de todos los ciudadanos ante la Sociedad de la Información, sino también como una oportunidad de negocio, dadas las dimensiones del mercado actual y, sobre todo, del futuro.

Tras la reciente aprobación de la Ley 7/2010, Ley General de la Comunicación Audiovisual [1], las cuotas de subtitulado de programas televisivos a las que están obligados los radiodifusores se sitúan en el 90 % para las televisiones públicas y el 75 % para las televisiones privadas. El subtitulado de programas que se emiten en diferido, como series, cine, documentales, y en general los programas grabados, es ya una realidad en España. Sin embargo el subtitulado de programas en directo, exceptuando los informativos, es aún una tarea en la que se debe realizar un esfuerzo de investigación para alcanzar mejores resultados y un menor coste. La necesidad de cumplir con las obligaciones establecidas por la Ley para 2013, obligará en los próximos años a implantar soluciones que ofrezcan a los usuarios un nivel de servicio con una calidad y costes adecuados en el subtitulado de programas en directo.

El proyecto pretende ser una solución real a estos problemas, de forma que el subtitulado pueda resultar viable económicamente hablando, además de sencillo. Pero no se queda ahí, sino que va más allá caracterizándose por dos conceptos clave, tanto para los proveedores de contenidos, como para los usuarios finales: automatización

y sincronismo. Y es que lo que este proyecto plantea es la posibilidad de obtener de forma automática, sin ningún tipo de intervención humana, un flujo de subtítulos completamente síncronos respecto al vídeo y el audio a los que hacen referencia. Este proyecto nace con la misión de desarrollar una de las principales partes del proyecto denominado como SISTEMA DE SUBTITULADO AUTOMÁTICO SINCRONIZADO en el que se hará uso de la tecnología ASR (Automatic speech recognition), lo que conllevará la necesidad de retardar un corto periodo de tiempo el programa a subtitular para así poder realizar la sincronización de manera adecuada [2][3][4].

Aunque la introducción de un retardo global que permita sincronizar los subtítulos pueda parecer un gran inconveniente, lo cierto es que en algunas arquitecturas de distribución de televisión no lo es, teniendo en cuenta las tecnologías de IPTV y el futuro inminente de la televisión: HbbTV (Hybrid Broadcast Broadband TV). Legalmente en España, no es posible retardar un programa emitido en directo en la señal broadcast, por lo que la solución pasa por generar un canal alternativo en el que el usuario interesado pueda seleccionar el subtítulo a pesar de que la señal llegue unos segundos retardada. No obstante, el ancho de banda del espectro radioeléctrico es muy limitado, por lo que la idea de emitir el mismo canal pero en versión subtitulada, resultaría complicada de asumir. Sin embargo, en IPTV no existen estas limitaciones. Es más, a medida que pasa el tiempo, las infraestructuras son mejores y el ancho de banda ofrecido por las redes, mayor, de ahí que la tendencia a integrar mayor número de servicios sobre la red, crezca continuamente tanto en número como en diversidad.

Así pues, estamos hablando de un escenario de aplicación actual, pero sobre todo futuro, en el que el subtitulado automático sincronizado sería sencillo, viable y económico, lo que puede resultar atractivo a muchos sectores de la Sociedad de la Información, pero principalmente al gran número de personas discapacitadas, las cuales serían las grandes beneficiadas del sistema que se plantea.

A lo largo del proyecto se detallará como el sistema es capaz de generar el flujo de subtítulos que posteriormente se multiplexarán junto con el audio/vídeo original de manera sincronizada con un ligero retraso respecto a la señal original en canales IPTV adicionales seleccionables por los usuarios.

1.2. Objetivos

El objetivo principal de este proyecto es la generación de un flujo de subtítulos DVB-SUB, ya sea mediante la codificación basada en píxeles o la basada en cadenas de caracteres, a partir del audio extraído de un canal de televisión. Este flujo debe ser generado de forma síncrona respecto a los flujos de vídeo y audio a los que hace referencia, de forma que posteriormente pueda reconstruirse un flujo de transporte integrando los flujos originales junto con el flujo de subtítulos sincronizados genera-

dos por el sistema. Dicho flujo de transporte podrá ser seleccionado por el usuario para reproducir una versión en casi-directo con subtítulos sincronizados de un programa de TV emitido en directo. Para ello, el proyecto ha de diseñarse teniendo en cuenta el sistema completo en el que se debe integrar, mencionado en la sección anterior como Sistema de subtitulado automático sincronizado. Este sistema estará compuesto de un elevado número de módulos, interconectados según las necesidades del sistema, por lo que entre los objetivos secundarios de este proyecto, y no por ello menos importantes, se encuentra satisfacer los requisitos derivados de la integración del módulo que este proyecto constituye, dentro del sistema completo.

Antes de proceder a la descripción de los requisitos y funcionalidad que debe cumplir este proyecto, es necesario realizar un estudio de las técnicas de subtitulado existentes en la actualidad, de la normativa vigente en materia de subtitulado para sistemas DVB, así como de otras técnicas necesarias para llevar a cabo los objetivos del proyecto. Esto va a resultar esencial para poder determinar las limitaciones y las posibilidades que existen a la hora de desarrollar un sistema de este tipo.

1.3. Estructura de la memoria

La presente memoria recoge la evolución del proyecto a lo largo de las siguientes secciones. Este documento está formado por siete partes y dieciséis capítulos.

La primera parte, en la que se incluye esta sección, se corresponde con el capítulo introductorio del proyecto.

La segunda parte, correspondiente al Estado del Arte, está compuesta por cuatro capítulos. El Capítulo 2, describe los aspectos más importantes a conocer de los sistemas de subtitulado en tiempo real. El Capítulo 3, comprende toda la normativa subyacente al proyecto, imprescindible para entender los aspectos a desarrollar en un sistema de este tipo. A continuación, en el Capítulo 4, realizaremos una descripción de las herramientas disponibles en la actualidad que puedan ser útiles para desplegar un sistema de subtitulado DVB. Para terminar con esta parte, se incluye el Capítulo 5, en el cual detallaremos las características más importantes del formato de imagen BMP.

Tras el análisis del Estado del Arte, en la tercera parte describiremos de forma concisa, los trabajos anteriores realizados en el Centro Español de Subtitulado y Audiodescripción [5] y que se se encuentran en estrecha relación con este proyecto. El Capítulo 6 describe el proyecto Apeinta, mientras que el Capítulo 7 comprende la explicación del proyecto Hermes-TDT.

A esta parte le seguirá una cuarta, en la que describiremos los trabajos realizados a lo largo del proyecto. Esta es la parte más extensa, y se compone a su vez de siete capítulos. En el Capítulo 8 se describirán los requisitos y la funcionalidad del sistema. A continuación, en el Capítulo 9, explicaremos la arquitectura del sistema. Tras esto,

en los Capítulos 10 y 11 describiremos el sistema de codificación de subtítulos DVB-SUB, basándonos en las dos alternativas posibles: la basada en píxeles y la basada en cadenas de caracteres. A esto le seguirá el Capítulo 12, donde se explicará el algoritmo de encapsulado en paquetes de transporte y el Capítulo 13, en el que describiremos las pruebas realizadas y los resultados obtenidos. Para terminar con esta parte, incluimos el Capítulo 14, en el que se describe la publicación de un artículo relativo al proyecto y el Capítulo 15, que contempla el Plan de Proyecto diseñado.

En la quinta parte de este documento, trataremos las conclusiones y las líneas futuras de investigación, recogidas en los Capítulos 15 y 16, respectivamente.

Acto seguido, el lector de este documento se encontrará con el glosario y la bibliografía y finalmente, en la séptima parte, se adjuntarán todos los anexos a los que se hacen referencia durante la memoria del proyecto.

1.4. Ámbito

El proyecto que a continuación se detalla, ha sido desarrollado dentro del marco de investigación del Centro Español de Subtitulado y Audiodescripción (CESyA) [5], cuyos trabajos en el campo de la accesibilidad a los medios audiovisuales tienen como objetivo el mejorar la integración social de personas con discapacidad y el acercamiento a la accesibilidad universal a través de la tecnología, principalmente en aplicaciones para la gestión de subtítulos o mecanismos de audiodescripción.

Dentro de este marco genérico, limitaremos nuestro ámbito de investigación a los subtítulos DVB emitidos por las cadenas de TV, tratando de mejorar las prestaciones que actualmente presentan, y de forma más concreta, con el principal objetivo de lograr una sincronización de los mismos respecto al contenido que se desea subtitular.

Parte II

Estado del Arte

Capítulo 2

Subtitulado en Tiempo real

2.1. Introducción

En el mundo actual, donde gran parte de la información y la comunicación sucede en un entorno audiovisual, es cada vez mayor el número de personas y situaciones que precisan de alguna ayuda tecnológica para acceder al contenido sonoro asociado a los eventos audiovisuales [8].

La trascripción del audio a texto puede ser necesaria para facilitar la comprensión no solo para personas con discapacidad auditiva o para aquellas con dificultad en la comprensión del idioma, sino para cualquier persona en ambientes con alto nivel de ruido. El subtitulado permite realizar una traslación, más o menos literal, de lo oral a lo escrito, y en la mayoría de los casos es la técnica más adecuada para facilitar el acceso al contenido audiovisual a esas personas. El subtitulado de los eventos que se presencian en tiempo real es una técnica de gran complejidad y coste elevado, por lo que está escasamente extendida en España y en los demás países occidentales.

A lo largo de este capítulo se describen los diferentes escenarios y tecnologías asociados a la subtitulación de eventos en tiempo real, así como una descripción de los tipos y formatos de subtítulos existentes.

2.2. Motivación

Según datos de la Encuesta sobre discapacidades realizada por el Instituto Nacional de Estadística (INE) en 2008 [6], el número de personas con algún tipo de discapacidad en España alcanza los 3,8 millones, lo que supone un 8,5 % del total de la población. De ellos, más de la mitad presentan deficiencias auditivas que van desde la sordera profunda a la mala audición. En el grupo de mayores de 80 años los problemas de audición alcanzan a un 18,2 % de los varones y a un 21,4 % de las mujeres. Todas esas personas están potencialmente excluidas de un alto número de eventos audiovisuales si no se proporciona algún tipo de ayuda, por lo que son los

principales beneficiados del subtitulado [8].

Otro de los públicos potenciales de los servicios de subtitulado son los inmigrantes residentes en España de habla no hispana, que pueden hacer uso de ellos para aprender el idioma. Según datos no oficiales de las emisoras de TV en España, solo la mitad de los usuarios habituales de subtitulado en las emisiones televisivas pertenecen al colectivo de usuarios con discapacidad auditiva. El resto lo hace por dificultades en la comprensión del idioma u otros motivos. Según datos del INE de la Encuesta de inmigrantes 2007 [7], en España residen 2.738.830 inmigrantes cuyo lugar de procedencia son países en los que el primer idioma no es el español. A esta cifra habría que añadir los procedentes de Brasil, ya que la base de datos del INE no permite desagregarlos del resto de países latinoamericanos.

2.3. Subtitulado

Subtitulado es el proceso de convertir a información visual, normalmente texto, la mayor parte posible de la información de audio de un evento audiovisual, comprendiendo el discurso hablado y/o parte de los efectos sonoros del evento. El subtitulado permite la comprensión de los eventos a las personas con discapacidad auditiva y a las que no dominan el idioma original, y puede emplearse además como alternativa del audio en situaciones con deficiencias acústicas.

Los subtítulos se presentan en forma de texto, a veces acompañados de algunas imágenes (e.g. emoticonos), de manera simultánea o casi simultánea a la parte audiovisual del evento. En la gran mayoría de los casos los subtítulos se presentan en pantalla superpuestos a la imagen. En otros casos, generalmente en situaciones en que el usuario está presente en el evento, los subtítulos se presentan en displays externos, ordenadores portátiles, dispositivos móviles, etc.

2.3.1. Tipos de subtítulos

Los subtítulos pueden ser clasificados según diferentes parámetros. A continuación se resaltan los distintos tipos de subtitulado según el criterio de clasificación: por audiencia, por idioma, por tiempo, por literalidad, por funcionalidad y por modo de distribución [9][8]. Además se establecerá la relación existente entre estas clasificaciones y el tipo de subtítulos que se da en DVB, siempre y cuando proceda establecer dicha relación.

2.3.1.1. Según la audiencia

Existen diferentes denominaciones para los subtítulos en función de la audiencia que tendrá acceso a ellos mientras se muestran:

- Subtitulado cerrado: Los subtítulos serán mostrados de manera que puedan ser vistos por una única persona, es decir, individualmente. Este método, permite ver los rótulos al espectador que lo desee o lo necesite, sin que lo perciban los demás espectadores[1].

- Subtitulado abierto: Los subtítulos son mostrados de modo que puedan ser vistos por varias personas de manera simultánea en una misma pantalla. En el caso de los subtítulos de DVB, el sistema abierto es el que se usa.

2.3.1.2. Según el idioma

Haciendo uso del idioma de los subtítulos como criterio de clasificación, se presentan dos categorías de subtítulos:

- Subtitulado interlingüístico: Es aquél en el que el idioma del audio original es diferente al idioma de los subtítulos asociados.

- Subtitulado intralingüístico: Es aquél en el que el idioma del audio original coincide con el de los subtítulos asociados.

En DVB se puede dar el caso de ambos, pues se pueden incluir varios flujos de subtítulos. No obstante, el caso habitual son los subtítulos intralingüísticos.

2.3.1.3. Según el tiempo

Es posible clasificar los subtítulos en base al momento en que se generan respecto al evento que subtitulan:

- Subtitulado en tiempo real: Es el que se crea en el momento en que el evento se está produciendo.

- Subtitulado diferido: Es el que se realiza en un momento temporal distinto al tiempo en que el contenido audiovisual fue creado.

2.3.1.4. Según la literalidad

Dependiendo de la correlación que exista entre el subtítulo y la información verbal del audio, se distinguen dos clasificaciones:

- Subtitulado literal: Aquel que transcribe la información verbal de manera literal o casi literal.

[1]Como parte de la actividad del CESyA, se desarrolló un sistema de subtitulado cerrado basado en tecnología HMD (*Head-mounted Display*). Es un sistema monocular consistente en un dispositivo de visualización montado sobre la estructura de unas gafas que permite al usuario visualizar los subtítulos directamente sobre uno de los ojos, permitiendo a la vez ver el contenido de la pantalla.

- Subtitulado adaptado: Es aquél en el que el texto expresa de manera equivalente el audio transcrito pero empleando palabras y expresiones más cortas o fáciles de leer en pantalla.

2.3.1.5. Según la funcionalidad

Se pueden clasificar los subtítulos en función de la información aportada por éstos al entorno audiovisual al que acompañan:

- Narrativos: Son aquellos que se corresponden con la transcripción del diálogo que se está llevando a cabo en el vídeo. Son los más comunes.

- Forzados: Aquellos que sólo aparecen para transcribir aquellos diálogos u otros elementos como textos que aparezcan en la obra y se encuentren en un idioma extranjero. Actualmente gozan de gran popularidad gracias la irrupción de las series extranjeras y a su facilidad de difusión a través de Internet.

- De contenido: Añaden información adicional a la obra, sustituyendo en muchas ocasiones a las propias imágenes.

- Informativos: Añaden información adicional a la obra pero sin formar parte de la misma.

- Para sordos: Muestran información adicional de sonidos no verbales que aporten información al espectador, como el sonido de un disparo o del ladrido de un perro.

2.3.1.6. Según el modo de distribución

Según el modo en que los subtítulos son proporcionados con el medio audiovisual, estos se pueden separar en:

- Incrustados o permanentes: Estos subtítulos se proporcionan ya insertados en la propia imagen a la que acompañan, de modo que no pueden ser separados de la misma. Su principal ventaja es, que al formar parte de dicha imagen, no es necesario el uso de ningún elemento adicional para su reproducción, evitando al reproductor la necesidad de renderizar el texto. No obstante, poseen diferentes desventajas:

  - No pueden ser desactivados.
  - No se pueden controlar parámetros como el tipo de fuente o color.
  - Omiten la posibilidad de utilizar más de una fuente de subtitulado, por ejemplo, para disponer de subtítulos en más de un idioma.

  - Su inserción en una imagen ya codificada anteriormente puede afectar negativamente a la calidad de la imagen, pudiendo aparecer artefactos alrededor del texto correspondiente al subtítulo.

- Pre-renderizados o flotantes: Se proporcionan de manera separada al stream del vídeo, aunque ya en formato de imagen. A diferencia de los incrustados, ofrecen la posibilidad de tener diferentes fuentes de subtitulado para un mismo vídeo, aunque siguen sin ofrecer capacidad para modificar sus parámetros de visualización (fuente, color, etc.). Exigen además la necesidad de que el reproductor sea compatible con su formato.

- Aislados: Son enviados en un stream alternativo al del vídeo original pero sin renderizar, es decir, como una serie de tiempos, textos y otras marcas para ser mostrados por el reproductor, que debe ser compatible con ellos. Son muy utilizados por su facilidad de creación y ofrecen una gran flexibilidad a la hora de su visualización, en tanto su formato de visualización puede ser variado durante la reproducción. Otro de los beneficios de este tipo de subtítulos es que su peso es mucho más pequeño que en el caso de los anteriores.

Teniendo en cuenta las características de DVB-SUB, tal y como se explicará en la sección 3.4 en la página 50, atendiendo a esta clasificación, nos encontramos ante subtítulos aislados. No obstante, es necesario resaltar algunas salvedades ya que no se puede aplicar de forma estricta esta clasificación para la codificación DVB-SUB basada en la codificación de píxeles. Mientras que DVB-SUB mediante codificación de caracteres cumple con las características del subtítulo aislado, en DVB-SUB mediante codificación de píxeles, en vez de textos, se mandan imágenes de subtítulos. Este aspecto hace que merme su flexibilidad a la hora de la visualización, dejando este aspecto en manos del proveedor del servicio, que ahora es quien determina cómo quiere que aparezca el subtítulo, en vez de que sea determinado por el reproductor.

2.3.2. Formatos de subtítulos

Existen multitud de formatos para enviar la información de subtitulado, adaptados a cada uno de los métodos de distribución existentes y con multitud de opciones en función del sistema al que estén destinados.

La información básica al almacenar un subtítulo serán los textos, o en su defecto, las imágenes, y sus marcas temporales. Sin embargo, algunos formatos permiten introducir más información como, por ejemplo, detalles sobre el estilo con el que se mostrarán los subtítulos, esto es, tipo de letra, color, etc.

Algunos de los formatos más comunes son [9]:

- EBU format (.stl)

- SubRip (.srt).
- MicroDVD (.sub).
- SubStation Alpha (.ssa) y Advanced SubStation alpha (.ass).
- Universal Subtitle Format (.usf).
- DVB-SUB.

El formato DVB-SUB difiere de los anteriores ya que carece de extensión. En realidad se trata de un flujo de datos que viaja dentro del flujo de transporte DVB codificando los subtítulos ya sea como cadenas de caracteres o como imágenes. El resto de tipos citados, son subtítulos en formato de texto plano. Dado el ámbito de este proyecto, nos vamos a centrar en los subtítulos con formato DVB-SUB, tanto en texto como en imagen.

2.4. Escenarios de subtitulado en tiempo real

Existen numerosos escenarios de comunicación donde el subtitulado constituye la principal ayuda para la comprensión. Aunque hace unos años la aplicación del subtitulado estaba restringida a una parte de ellos, hoy día son cada vez más las situaciones en las que es posible ofrecer esta ayuda.

Teniendo en cuenta las implicaciones tecnológicas, es relevante clasificar los escenarios en dos grandes grupos: los de subtitulado en diferido y los de subtitulado en tiempo real.

Los escenarios de subtitulado en diferido son todos aquellos escenarios en los que el evento audiovisual y el acto de subtitulado suceden en distintos momentos del tiempo. En estos casos, la preparación de los subtítulos se hace con antelación a la difusión del evento y no existen por tanto exigencias de tiempo real ni simultaneidad. La tecnología requerida en estos casos es mucho más simple y permite actualmente ofrecer subtítulos en todos los eventos audiovisuales que se presencien en diferido. Ejemplos de escenarios de subtitulado en diferido son la proyección de películas de cine, películas de TV, series, documentales, programas de TV en redifusión, etc.

Los escenarios de subtitulado en tiempo real son todos aquellos escenarios en los que el evento audiovisual y el acto de subtitulado suceden simultáneamente o casi simultáneamente. En estos casos, la creación y presentación de los subtítulos debe suceder de manera síncrona, o lo más síncrona posible, con el audio del evento. Ejemplos de escenarios de subtitulado en tiempo real son los informativos de TV, programas de TV en directo, conferencias, tertulias en TV, debates en TV, debates parlamentarios, clases, retransmisión de eventos deportivos, teatro, óperas, etc.

En el estado del arte actual para la generación de subtítulos en directo, hay siempre un retardo significativo entre el audio y el momento en que los subtítulos

	Audiencia	Idioma	Tiempo	Literalidad	Funcionalidad
Informativos	Cerrados	Intraling.	Tiempo real/Semi-directo	Literales	Narrativos
Conferencias	Cerrados/ Abiertos	Intra/ Interling.	Tiempo real	Adaptados/ Literales	Narrativos
Clases	Cerrados	Intraling.	Tiempo real	Literales	Narrativos
Debates TV	Cerrados/ Abiertos	Intraling.	Tiempo real	Adaptados/ Literales	Cerrados
Deportes TV	Cerrados	Intraling.	Tiempo real	Adaptados/ Literales	Narrativos
Conv. telef.	Cerrados	Intraling.	Tiempo real	Adaptados/ Literales	Narrativos

Tabla 2.1: Tipos de subtitulado requeridos en escenarios audiovisuales de tiempo real.

correspondientes están disponibles. Esta es la razón por la que los subtítulos aparecen en el equipo de recepción del usuario un número variable de segundos más tarde que los fragmentos originales de audio original. Este retardo puede dificultar de manera considerable la comprensión [10].

Una gran parte de los escenarios audiovisuales que requieren subtitulado en tiempo real son programas televisivos de distinto tipo y con distintas dificultades a la hora de subtitular. Hasta hace poco tiempo apenas dos cadenas de televisión subtitulaban eventos en estricto directo, aunque hoy día, el subtitulado en modo semi-directo (basado en los textos previamente preparados por los equipos de redacción de noticias) está disponible en la práctica totalidad de los informativos de las cadenas en emisión nacional y autonómica.

La tabla 2.1 muestra los distintos tipos de subtitulado que se emplean con más frecuencia en algunos escenarios de subtitulado en tiempo real.

2.5. Tecnologías de subtitulado en tiempo real

De acuerdo a la sección 2.4, hemos hecho distinción de dos escenarios de subtitulado: subtitulado en diferido y subtitulado en tiempo real. Como paso previo a la descripción de las tecnologías existentes en los escenarios de subtitulado en tiempo real, es conveniente hacer mención del método clásico de subtitulado que ha existido en el subtitulado en diferido. Este método consiste en la transcripción manual de aquello que se quiere representar en el subtitulo. Se trata de un proceso lento y tedioso, que no obstante, proporciona un índice de errores muy bajo [9].

Respecto a los escenarios de subtitulado en tiempo real, el elemento clave es la trascripción inmediata del discurso de uno o varios hablantes. Los principales subsistemas funcionales del subtitulado en tiempo real son la trascripción de voz a texto, la generación de subtítulos y la transmisión/presentación de los subtítulos, tal como se muestra en la Figura 2.1.

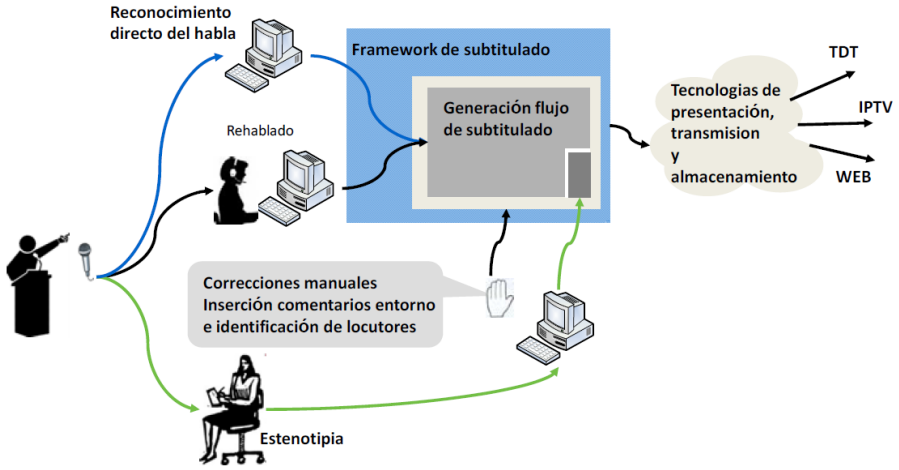


Figura 2.1: Arquitectura genérica de un sistema de subtitulado en tiempo real[10].

La trascripción de voz a texto se puede realizar de dos formas principalmente: mediante sistemas de teclado y estenotipia computerizada y mediante sistemas de reconocimiento automático del habla (ASR[2]).

2.5.1. Estenotipia Computerizada

La estenotipia computerizada es una técnica basada en la transcripción por estenotipia de una persona entrenada que es capaz de teclear en una máquina de estenotipia a la misma velocidad del discurso.

El estenotipista recoge los signos de puntuación y la información sonora relevante (aplausos, ruidos, risas, entonaciones...). La máquina de estenotipia, con un reducido número de teclas (21, la máquina de tipo Grandjean y 23, la maquina del sistema Melani), permite transcribir sílabas o palabras, mientras que un ordenador asociado genera el subtitulado final.

Los principales problemas de este método de trascripción de voz y audio a texto son la disponibilidad y el coste elevado debido la escasez de profesionales y a la rotación necesaria para cubrir eventos de duración media o larga. Es por esto por lo que no es una alternativa conveniente para el subtitulado masivo en televisión. En la Figura 2.2 se puede observar un ejemplo de máquina de estenotipia, en concreto, una del sistema Melani.

[2]Automatic speech recognition

Figura 2.2: Máquina de estenotipia.

2.5.2. Sistemas de reconocimiento automático del habla (ASR)

"El reconocimiento automático del habla es el procedimiento por el cual se convierte una señal acústica, capturada por un micrófono, en un conjunto de símbolos de un diccionario dado. Dichos símbolos están generalmente asociados con elementos semánticos de tipo palabra"[3].

Las tecnologías de reconocimiento del habla son la base para la viabilidad del subtitulado masivo de eventos audiovisuales en directo. El avance en los últimos 5 años en las tecnologías de ASR está permitiendo la introducción paulatina de sistemas de ASR para transcribir a texto o subtitular informativos de televisión, clases, discursos, debates televisivos, conferencias, etc. si bien en muchos de los escenarios se hace con un rehablador interpuesto.

Sin embargo, las prestaciones del mejor sistema construido en la actualidad, en lo relativo a las tasas de reconocimiento, están muy por debajo respecto a las que serían atribuibles al ser humano y, por consiguiente, aún lejos de proporcionar resultados de trascripción aceptables para los usuarios de un servicio como el subtitulado. La solución global requiere la incorporación de conocimientos multidisciplinares, incluyendo trabajos de investigación básica en los campos de producción y percepción del habla.

Las etapas principales de un sistemas de reconocimiento del habla son:

1. El procesamiento o análisis del habla, donde se usan técnicas de Fourier (STFT) o Filtrado óptimo probabilístico (POF).

2. La clasificación de unidades fonéticas o modelo acústico, donde se usan técnicas Hidden Markov Model (HMM) o Redes de neuronas.

[3]R. A. Cole et al., Survey of the State of the Art in Human Language Technology, Cambridge University Press, 1997

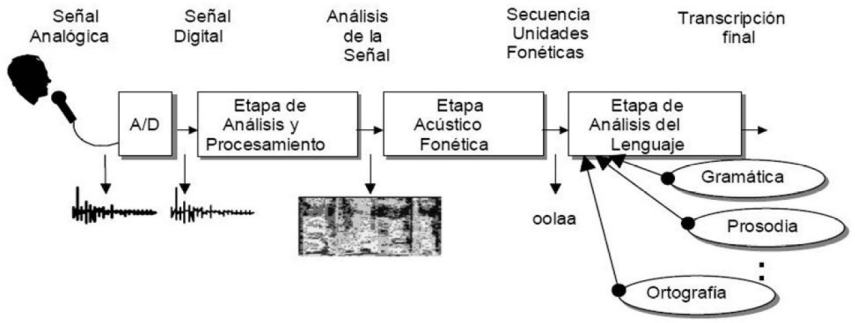


Figura 2.3: Etapas de un sistema de reconocimiento de habla (ASR)[8].

3. Análisis en función de reglas del lenguaje o modelo del lenguaje, por medio de la técnica de incorporación de información prosódica (de acentuación) a un sistema de ASR basado en HMM.

Existen aspectos en la utilización de los sistemas ASR que condicionan y determinan su uso para la trascripción de voz a texto. Por un lado, el aprendizaje y entrenamiento son factores clave: el tono de voz, la entonación, la claridad en la pronunciación, la personalización de comandos y la creación y mantenimiento de vocabulario específico. Por otro lado, la obtención de resultados óptimos requiere la corrección posterior al dictado, con los costes de aprendizaje y cognitivos que supone la corrección de los errores, lo que hace que en ocasiones la productividad sea similar al uso de teclado y ratón pero con más errores. Otro factor a tener en cuenta es que no todos los sistemas proporcionan reconocimiento del hablante, lo que introduce una carga adicional debido a la necesidad de introducir, casi siempre manualmente, información que ayude a reconocer los fragmentos del texto que corresponden a cada hablante. Por último, existen problemas técnicos relevantes, como son la gran potencia de cálculo que requieren los sistemas ASR, el retardo introducido en la generación de las transcripciones, la saturación del sistema, y la compatibilidad e integración del ASR con otros programas informáticos.

La tabla 2.2 muestra los elementos que caracterizan el comportamiento de un sistema ASR. En algunos casos los rangos son tan amplios que es difícil asegurar el funcionamiento correcto en todas las situaciones. De ahí que la mayoría de los progresos en las técnicas ASR, y su aplicación a escenarios particulares, tenga lugar en situaciones específicas, o muy específicas, bajo las cuales es posible asegurar comportamientos aceptables. Asegurarlos en todas las circunstancias, como se ha mencionado, es una meta aún lejana.

No obstante, la mejora de los sistemas de ASR es un hecho, que ha llevado más de 50 años alcanzar. La complejidad de esta disciplina hace que en la práctica exista solo un número reducido de *kernel* y distintas adaptaciones optimizadas para aplicaciones específicas. Algunos de los sistemas de ASR comerciales son Dragon NaturallySpeaking, ViaVoice, ViaScribe, Media Mining Indexer, etc. A día de hoy,

Parámetro	Rango
Modo de Hablado	Palabras aisladas - Discurso continuo
Estilo de Hablado	Discurso leído - Discurso espontáneo
Perfil	Dependiente del hablante - Independiente del hablante
Vocabulario	Pequeño (<20 palabras) - Grande (>50.000 palabras)
Modelo de lenguaje	Estados finitos - Sensible al contexto
Grado de Confusión	Pequeño (<10 palabras) - Grande (>200 palabras)
SNR	Alta (>30dB) - Baja (<10dB)
Transductor	Micrófono cancelador de ruido - Auricular de teléfono

Tabla 2.2: Variables que caracterizan un sistema ASR[8].

las dos más importantes son Dragon NaturallySpeaking [11] de Nuance, que es el reconocedor utilizado en el Sistema de Subtitulado en Directo para Educación del CESyA [5], y por otro lado, ViaVoice, desarrollado por IBM. Las tasas de éxito típicas en los sistemas ASR para diversos tipos de entrada son mostradas en la tabla 2.3 en la página siguiente.

Sin embargo, los sistemas de ASR presentan una serie de problemas en relación al subtitulado que impiden hoy en día una utilización no asistida de los mismos. Algunos de estos problemas son los siguientes:

- El retardo que introduce el sistema ASR entre el audio y vídeo originales y el subtitulado asociado[4].
- La consecución de un alto porcentaje de acierto requiere una elevada potencia de cálculo si se desea minimizar el retardo, y una considerable inversión de tiempo para entrenar el sistema con el perfil de voz del locutor.
- La tasa de aciertos en sistemas no entrenados o con un alto nivel de ruido está por debajo del nivel requerido para el subtitulado en tiempo real.
- La transcripción obtenida mediante los ASR necesita ser corregida (errores, signos de puntuación, etc). Además, es necesario asignar identificación del locutor a los distintos fragmentos del discurso y añadir información de otros efectos sonoros relevantes.

2.5.2.1. El rehablado

El rehablado es una técnica que permite realizar la trascripción de voz a texto mediante un sistema de ASR, y donde el discurso original es repetido, de manera

[4]Este es uno de los principales problemas que se aborda en este proyecto.

Corpus	Tipo de hablado	Tamaño del léxico	Tasa de error de palabra	Tasa de error humana
Cadenas de dígitos	Espontáneo	10	0,3	0,009
Gestión de recursos	Lectura	1.000	3,6	0,1
ATIS	Espontáneo	2.000	2	–
Wall Street Journal	Lectura	64.000	6,6	1
Noticias de radio	Mixto	64.000	13,5	–
Centralita telefónica	Conversación	10.000	19,3	4
Llamada telefónica doméstica	Conversación	10.000	30	–

Tabla 2.3: Tasas de éxito en los sistema ASR[8].

lo más fidedigna posible y en un entorno de condiciones acústicas óptimas, por una persona con formación específica y que ha entrenado previamente el sistema ASR, de tal manera que se reduzca al máximo el número de errores generados en el reconocimiento. El rehablado es una forma práctica de aprovechar las capacidades de los sistemas ASR actuales.

La característica más importante del rehablado es el esfuerzo requerido en la dicción y en la síntesis del discurso original, siendo sus principales desventajas las siguientes:

- Es una técnica que precisa de gran concentración.
- Requiere la formación de los rehabladores.
- Es preciso un entorno exento de ruidos y controlado, provisto de los medios suficientes para entregar un sonido limpio.
- El coste es elevado.
- El subtitulado de eventos con más de un locutor requiere generalmente de más de un rehablador.

El rehablado es una técnica ampliamente utilizada hoy en día para el subtitulado de programas televisivos en directo. Aunque el liderazgo lo ostenta la cadena BBC

en Reino Unido, con cerca del 100 % de la programación subtitulada, existen cada vez más iniciativas para subtitular mediante rehablado los programas en directo de mayor interés. En España, el rehablado es la técnica principal en el subtitulado masivo de programas televisivos en directo, hasta ahora solo disponibles en los canales de RTVE.

2.5.3. *Framework* de subtitulado

Aunque los sistemas de ASR asistido por rehablado pueden proporcionar tasas de aciertos cercanas al 90 %, el subtitulado de eventos audiovisuales en tiempo real precisa de una etapa adicional de generación de subtítulos, donde el texto que entrega el sistema ASR es manipulado con el fin de darle la forma definitiva antes de ser difundido, transformando el texto en subtítulos [8].

Los sistemas de generación de subtítulos que soportan el subtitulado de eventos en tiempo real se conocen como *framework* de subtitulado o *live subtitling*. Los frameworks de subtitulado requieren de un operador que en la mayoría de los casos reparte su actividad entre el rehablado y las funciones del live subtitling. En situaciones audiovisuales de alta complejidad puede requerirse un número mayor de personas para poder responder en tiempo real a la demanda del sistema.

Las tareas a cargo de la persona que maneja el framework de subtitulado son las siguientes:

- Corrección de los errores del ASR.
- Introducción de signos de puntuación.
- División del texto en sentencias de subtitulado.
- Asegurar que la longitud, cadencia y tiempo de permanencia de los subtítulos permiten su lectura (según la guía de estilo o normativa aplicable).
- Sincronización de subtítulos (en algunos escenarios).
- Introducción de claves (generalmente colores) para la identificación de personajes.
- Introducción de símbolos y texto que identifiquen elementos sonoros no reconocidos en la etapa ASR: risas, aplausos, sonidos relevantes, etc.

En Europa, los principales framework de subtitulado comerciales son Fab Subtitler, Softel, Cavena Tempo y Anglatècnic Fingertext [12]. Se realizará una descripción detallada de los tres primeros en la sección 4.4[5].

[5]Nos centramos en esos tres, ya que el sistema de Anglatécnic Fingertext está basado en teletexto en vez de en DVB-SUB.

Capítulo 3

Normativa vigente en DVB-T y DVB-SUB

3.1. Introducción

El proyecto DVB fue fundado en Europa en 1993 como un consorcio formado por 290 cadenas de televisión, fabricantes, operadores de red, desarrolladores de software, órganos regulatorios y otros, en un total de 35 países, con el propósito de crear y proponer los procedimientos de estandarización para la televisión digital compatible [13].

Desde sus inicios, son muchos los estándares y recomendaciones propuestos por DVB, cubriendo así todos los aspectos relativos a la televisión digital, por lo que es importante delimitar adecuadamente los aspectos que se pretenden cubrir a lo largo del proyecto. De esta forma, evitaremos navegar a la deriva entre el mar de estándares propuestos por DVB para la televisión digital.

Así pues, vamos a centrar nuestra atención en el estándar DVB-T, que es el que comprende la televisión digital a través de redes de distribución terrestres utilizando los canales UHF convencionales, y de forma más concreta, en el sistema de subtitulado de DVB. Con este fin, a continuación se van a explicar los tres estándares necesarios para entender cómo desplegar un sistema de subtitulado DVB en un canal TDT.

- *Generic coding of moving pictures and associated audio information: Systems. ISO/IEC 13818-1*: comúnmente conocido como *MPEG-2 Systems*, define la forma de multiplexar uno o más flujos elementales, ya sean de vídeo, audio o datos, con el fin de crear uno o varios flujos susceptibles de ser almacenados o transmitidos. Para ello se definen dos tipos de flujos, optimizados para diferentes tipos de aplicaciones: ***Program Stream*** (PS), para entornos casi libres de errores, y ***Transport Stream*** (TS), para entornos ruidosos [14].

- *Digital Video Broadcasting (DVB); Specification for Service Information (SI)*

in DVB systems. ETSI EN 300 468 [15]: este documento especifica los datos de información de servicio que forman parte de los flujos de DVB de tal forma que el usuario sea provisto de la información necesaria para seleccionar los servicios o eventos que existen dentro del flujo y que el receptor IRD pueda configurarse automáticamente para el servicio seleccionado. Los datos SI para la configuración automática se encuentran explicados en su mayor parte dentro de "*ISO/IEC 13818-1"* [14] como *Program Specific Information (PSI).*

- *Digital Video Broadcasting (DVB); Subtitling Systems. ETSI EN 300 743* [16]: este documento especifica el método por el que subtítulos, logos y otros elementos gráficos pueden ser codificados y transportados dentro de los flujos DVB. Para ello, el sistema define *Colour Look-Up Tables* (CLUTs), es decir, tablas que definen los colores de los elementos gráficos. El transporte de los elementos gráficos codificados está basado en el documento "*MPEG-2 Systems" [14].*

Por último, vamos a describir las características del sistema de subtítulos en formato Teletexto (*Digital Video Broadcasting (DVB); Specification for conveying ITU-R System B Teletext in DVB bitstreams. ETSI EN 300 472*), que aunque no se encuentra dentro del ámbito del proyecto, ya que este sistema no ha sido implementado, es importante tenerlo en cuenta, puesto que, a pesar de ser un sistema atrasado tecnológicamente respecto a la norma DVB-SUB, está aún presente en muchas de las cadenas de televisión.

3.2. Generic coding of moving pictures and associated audio information: Systems. ISO/IEC 13818-1

Esta recomendación/estándar internacional especifica la capa de sistema correspondiente a la codificación MPEG-2. Fue desarrollado principalmente para permitir la combinación de vídeo y audio codificados mediante los métodos definidos en las partes 2 y 3 de este estándar. Asimismo, contempla la posibilidad de incluir datos relativos a la emisión, como pueden ser los subtítulos. La capa de sistema permite cinco funciones básicas:

1. La sincronización de múltiples flujos comprimidos, durante el proceso de decodificación.

2. El intercalado de múltiples flujos comprimidos en un único flujo.

3. La inicialización del buffer durante el arranque de la decodificación.

4. La gestión continua del buffer.

5. La identificación de tiempos.

La codificación a nivel de sistema, tal y como ya hemos mencionado durante la introducción, se debe realizar de dos formas: en forma de Transport Stream o en Program Stream. La definición de ambos tipos de flujos que se hace en este estándar determina la sintaxis necesaria y suficiente para la decodificación y presentación de la información de vídeo y audio, a la vez que asegura que los buffers de los decodificadores no se desbordarán o se vaciarán por completo en ningún momento. La información codificada contiene los dos tipos de marcas temporales que se muestran a continuación:

- Marcas temporales concernientes a la decodificación y presentación del audio codificado y los datos visuales.

- Marcas temporales relativas a la entrega de los datos del flujo en sí.

Ambas definiciones de flujos están orientadas a la multiplexación de paquetes, que reciben el nombre de ***Packetized elementary streams*** (PES). Para entender este mecanismo de multiplexación, vamos a poner un ejemplo básico en el que se pretende multiplexar dos flujos elementales: uno de vídeo y otro de audio. Cada uno de estos flujos elementales se compone de paquetes PES que se multiplexan tal y como se puede observar en la Figura 3.1 en la página siguiente. En ella se explica de forma genérica cómo generar tanto un PS como un TS a partir de la misma fuente de datos.

A partir de ahora nos centraremos en los procesos que ocurren en el lado derecho de la línea punteada de la figura, pues son los que cubre la norma ISO/IEC 13818-1. En primer lugar, explicaremos de forma concisa los aspectos más relevantes de MPEG-2 Systems, para a continuación centrar nuestra atención en la estructura de los Transport Stream, pues es el formato usado para transmitir la información en DVB.

3.2.1. Program Stream

Un Program Stream o flujo de programa, resulta de la combinación de uno o más flujos de paquetes PES, cuya base de tiempos es común, en un único flujo. Está diseñado para transmitir o almacenar **un único programa** de datos codificados (audio y/o vídeo) u otro tipo de datos en entornos donde los errores son muy poco probables. Por esta razón, los paquetes de un PS pueden ser de longitud relativamente grande, ya que no existe peligro de pérdidas de información.

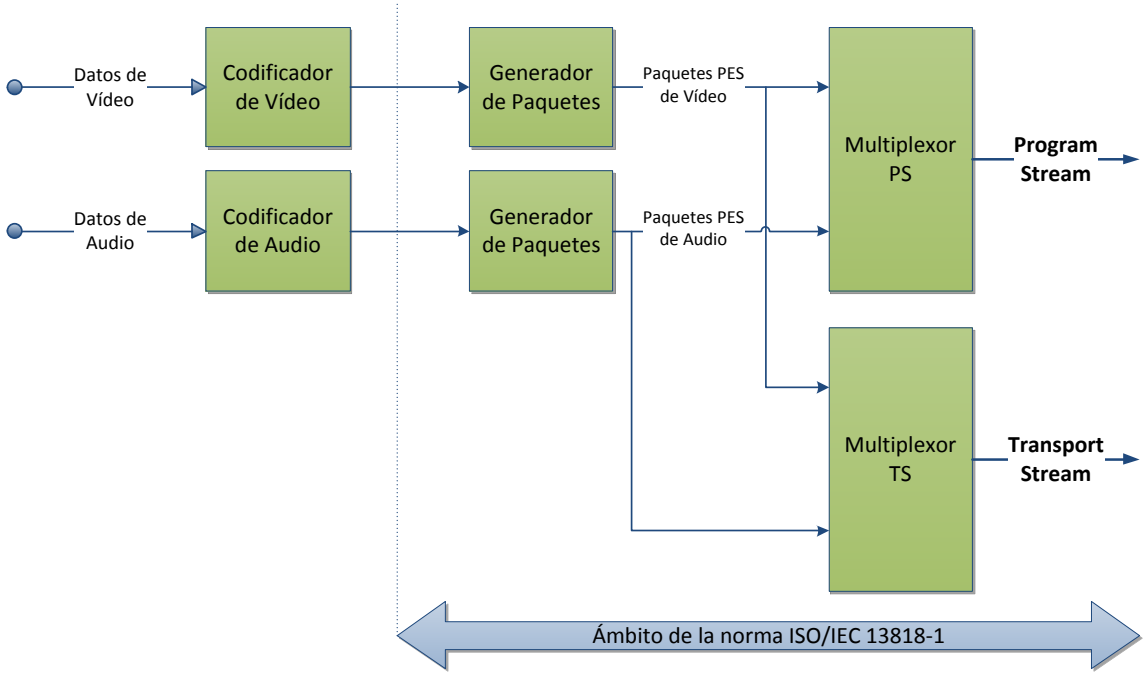


Figura 3.1: Mecanismo de multiplexación en MPEG-2 [14].

Los flujos de programa pueden ser de tasa fija o tasa variable, al igual que los flujos elementales que lo conforman. La tasa del PS queda definida por los valores y localización del *System Clock Reference* (SCR) y del *program_mux_rate*[1].

Un flujo de programa se compone de dos capas: la capa de sistema y la capa de compresión. A su vez, la capa de sistema se divide en dos subcapas: la de operaciones *multiplex-wide* y la de operaciones específicas de flujo (capa de paquetes PES). El flujo de entrada al decodificador de PS presenta una capa de sistema que envuelve a la capa de compresión. Esta capa es interpretada por el decodificador de forma que los flujos de entrada que llegan a los decodificadores de vídeo y de audio, solo se componen de la capa de compresión. En la Figura 3.2 en la página siguiente, se presenta un prototipo de decodificador de flujos de programa indicando en cada momento las capas que permanecen en cada momento de la decodificación.

3.2.2. Transport Stream

La definición de Transport Stream o flujo de transporte fue diseñada con el fin de transmitir o almacenar **uno o varios programas** de datos codificados de acuerdo a las normas ITU-T Rec. H.262 | ISO/IEC 13818-2 (datos de vídeo) e ISO/IEC 13818-3 (datos de audio), así como otros tipos de datos, en entornos susceptibles de sufrir errores. Dichos errores se pueden manifestar en forma de alteraciones en

[1]Para obtener más información sobre estos campos, revisar la información concerniente a ellos en ISO/IEC 13818-1 [14]

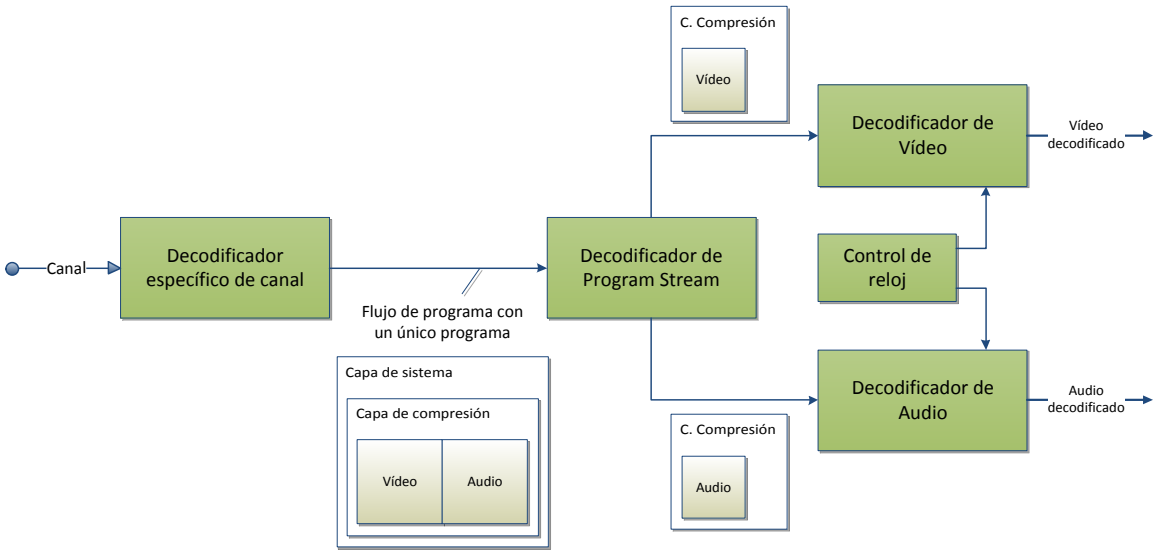


Figura 3.2: Modelo de decodificación de un PS [14].

los valores de los bits o en pérdidas de paquetes. Debido a esto, los paquetes de un flujo de transporte son de tamaño fijo, 188 bytes, de los cuales los 4 primeros son de cabecera y el resto son datos procedentes de un único paquete PES *(dentro de un paquete de transporte no puede haber datos de diferentes PES).*

Al igual que los flujos de programa, los flujos de transporte pueden ser de tasa fija o variable, así como sus respectivos flujos elementales. Sin embargo, la definición de la tasa del flujo de transporte se realiza mediante los valores y localizaciones de los campos *Program Clock Reference* (PCR) en vez de mediante los SCR. En general, los PCR son diferentes para cada programa.

La forma de usar el PCR es la siguiente: el tiempo t(i) en el que el i-ésimo byte entra en el decodificador, se define mediante la decodificación del PCR del flujo de entrada, codificado en el campo de adaptación del paquete de transporte a decodificar (el campo de adaptación se explica en la subsección 3.2.7.2) y a través de la cuenta de los bytes que hay en el flujo de transporte entre PCRs sucesivos del mismo programa. El PCR (ver ecuación 3.1) es un campo de 42 bits codificado en dos partes:

1. *program_clock_reference_base*: campo codificado en unidades de periodo de 1/300 veces la frecuencia del reloj de sistema. El valor codificado se calcula según indica la fórmula 3.2.

2. *program_clock_reference_extension*: campo codificado en unidades de frecuencia del reloj de sistema. El valor codificado se calcula según indica la fórmula 3.3.

Así pues, el valor codificado en el PCR indica el tiempo t(i) en el que el byte i-ésimo

contiene el último bit del campo program_clock_reference_base.

$$PCR\,(i) = PCR\_base\,(i) \times 300 + PCR\_ext\,(i) \tag{3.1}$$

$$PCR\_base\,(i) = \{[system\_clock\_frequency \times t\,(i)]\,/300\}\ \%2^{33} \tag{3.2}$$

$$PCR\_ext\,(i) = \{[system\_clock\_frequency \times t\,(i)]\,/1\}\ \%300 \tag{3.3}$$

Las posibilidades existentes para construir un flujo de transporte son muy variadas, siendo válidas todas aquellas que desemboquen en un flujo bien formado. Es posible construir un flujo de transporte conteniendo uno o varios programas cuyos datos procedan de las siguientes fuentes de datos:

- Flujos elementales de datos codificados.
- Otros flujos de programa.
- Otros flujos de transporte que a su vez pueden contener uno o varios programas.

Debido a esto, el flujo de transporte fue diseñado de tal forma que se pudieran realizar determinadas operaciones de una forma sencilla y mediante el mínimo esfuerzo. Entre estas operaciones están las siguientes:

- Obtener los datos codificados de un programa perteneciente a un flujo de transporte, decodificarlos y presentar los resultados tal y como se muestra en la Figura 3.3 en la página siguiente.
- Extraer los paquetes de un flujo de transporte pertenecientes a un programa y generar como salida un flujo de transporte con ese único programa.
- Extraer los paquetes de uno o varios programas contenidos en uno o varios flujos de transporte y producir como salida un flujo de transporte totalmente diferente con la mezcla de dichos programas.
- Extraer los contenidos de un programa procedente de un flujo de transporte y generar como salida un flujo de programa conteniendo dicho programa.
- Convertir un PS en un TS con el fin de ser transportado por un medio con pérdidas, para posteriormente ser recuperado como un flujo de programa válido y en algunos casos, idéntico al original.

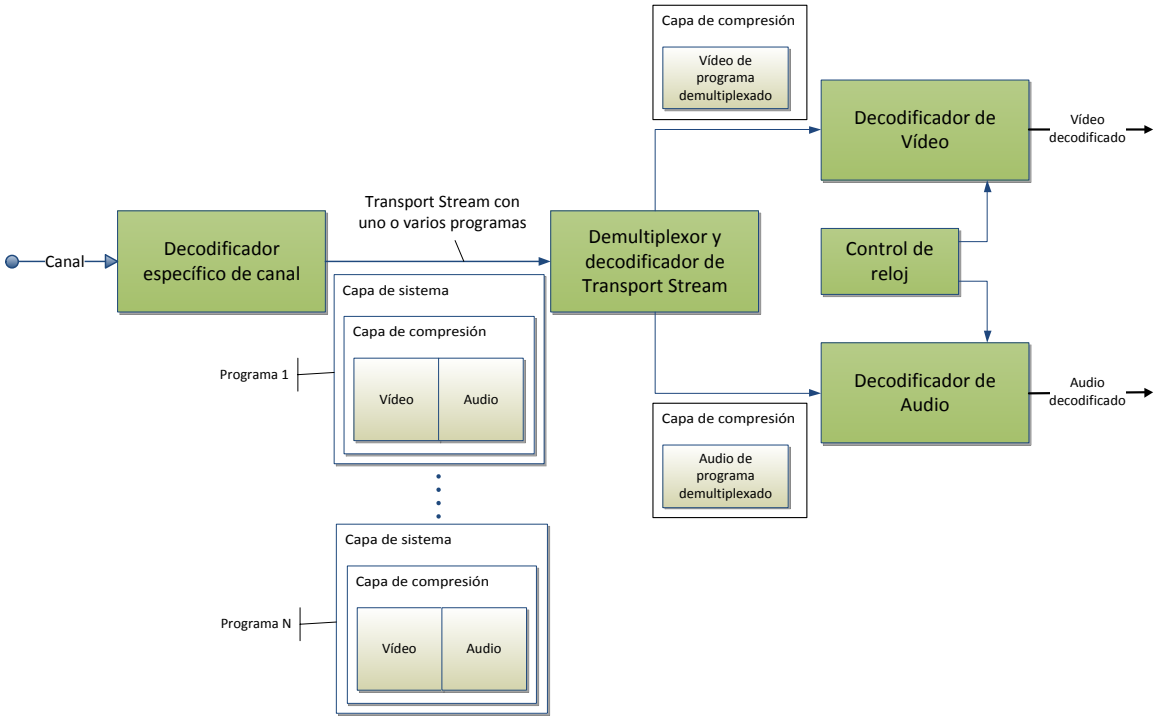


Figura 3.3: Modelo de decodificación de un TS [14].

De forma análoga a los flujos de programa, los flujos de transporte se componen de dos capas: la capa de sistema y la capa de compresión. De igual manera, la capa de sistema queda subdividida en las mismas dos subcapas: la de operaciones multiplex-wide (ver subsección 3.2.4) y la de operaciones específicas de flujo (ver subsección 3.2.5).

3.2.3. Packetized Elementary Streams

Constituyen el resultado del proceso de generación de paquetes, tal y como se puede observar en la Figura 3.1, por lo que son la base lógica de los flujos de transporte y de programa. El hecho de ser la misma base para ambos tipos de flujos, permite que se pueda realizar la conversión entre flujos de transporte y de programa y viceversa. Se dice que son las unidades lógicas que componen los flujos pues representan la unidad mínima presentable. Por ejemplo, un paquete PES podría ser un frame de vídeo, es decir una porción de información que podría ser interpretada por un decodificador y presentada en pantalla. En cambio, las unidades físicas, son los paquetes de transporte o paquetes de programa, que tal y como hemos mencionado, tienen sentido a nivel de sistema y constituyen la forma de transmitir la información o de almacenarla. En casos como los flujos de transporte, los paquetes PES pueden ser mucho más largos que un paquete de transporte, ya que, como ya se ha mencionado, son siempre de 188 bytes.

La mejor forma de entender el significado de un paquete PES es mediante la Figura 3.4. En ella se puede observar como a partir de un flujo elemental del tipo que sea (vídeo, audio o datos), se generan diferentes unidades lógicas, o lo que es lo mismo, se genera una secuencia de paquetes PES. Todos los paquetes PES pertenecientes al mismo flujo elemental presentan el mismo *Stream ID* (Identificador de flujo), lo que permite identificar en todo momento el origen de los PES. A continuación explicamos los campos más importantes de la cabecera:

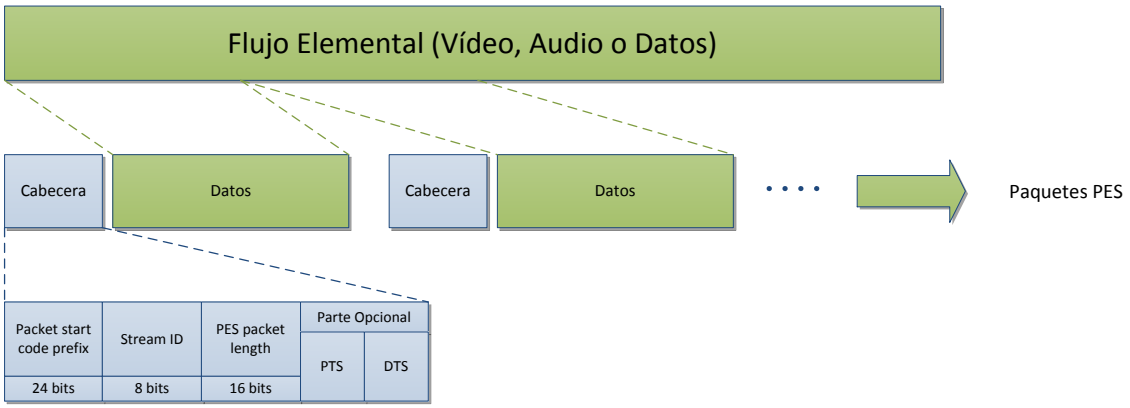


Figura 3.4: PES y flujos elementales.

- *Packet start code prefix*: es un código de 24 bits que, junto con el Stream ID, identifica el inicio de un paquete PES. El valor de este código en hexadecimal es:

$$packet\_start\_code\_prefix = 0x000001$$

- *Stream ID*: es un campo de 8 bits que especifica el tipo de flujo elemental del que procede la información. En el caso de los flujos de programa, además indica el número del flujo elemental. Los valores permitidos en este campo están contemplados en la tabla 2-18 de ISO/IEC 13818-1 [14].

- *PES packet length*: campo de 16 bits que indica el número de bytes en el paquete PES a partir de este campo.

- *PTS (presentation time stamp)*: es un número de 33 bits, separado en tres campos, que indica el **tiempo de presentación** en el decodificador, $tp_n(k)$, de una unidad de presentación $P_n(k)$ correspondiente a un flujo elemental determinado. El valor del PTS se especifica en unidades del periodo del reloj del sistema dividido entre 300 (lo que da lugar a incrementos de 90 KHz). El PTS se calcula tal y como se indica en la ecuación 3.4, donde la frecuencia del reloj de sistema es $27MHz$.

$$PTS\,(k) = \{[system\_clock\_frequency \times tp_n\,(k)]\,/300\}\ \%2^{33} \qquad (3.4)$$

- *DTS (decoding time stamp)*: es un número de 33 bits separado en tres campos que indica el tiempo de decodificación en el decodificador, $td_n(j)$, de una unidad de acceso $A_n(j)$ correspondiente a un flujo elemental determinado. Una unidad de acceso es la representación codificada de una unidad de presentación, es decir, constituye la información antes de ser decodificada. Al igual que el PTS, el valor del DTS es especificado en unidades del periodo del reloj del sistema dividido entre 300. El DTS se calcula tal y como se indica en la ecuación 3.5.

$$DTS\,(j) = \{[system\_clock\_frequency \times td_n\,(j)]\,/300\}\ \%2^{33} \tag{3.5}$$

En el caso de unidades de presentación que no requieran corregir el retardo acumulado, $tp_n(k)$ y $td_n(j)$ coinciden ya que las unidades de acceso son decodificadas de manera instantánea, por lo que no es necesario incluir el DTS[2]. En cambio, para las unidades de acceso que deban ser retardadas, $tp_n(k)$ y $td_n(j)$ difieren en el tiempo en que la $P_n(k)$ es retardada en el buffer de reordenación.

3.2.4. Operaciones *Multiplex-wide*

Tal y como se mencionó anteriormente, constituye una de las subcapas que forman la capa de sistema. Dentro de esta subcapa se incluyen operaciones de diversa naturaleza como la coordinación de la adquisición de datos de un canal, el ajuste de los relojes o el manejo de los buffers de los decodificadores. Todas estas tareas están íntimamente relacionadas de forma que unas dependen de otras. Así por ejemplo, el hecho de poder controlar la tasa con la que los datos son entregados, permite asegurarse de que los buffers nunca se vacían o llenan al completo.

Otra de las operaciones incluidas en esta subcapa es la habilidad para determinar qué recursos son necesarios para decodificar un flujo de transporte o de programa. Para ello, se hace uso de la información contenida en las cabeceras de los paquetes, que permite entre otras cosas, identificar las características y relaciones existentes entre los diferentes flujos elementales que componen cada programa. Un ejemplo de esta información puede ser incluir el lenguaje hablado en cada canal de audio.

3.2.5. Operaciones específicas de flujo - Capa de paquetes PES

La segunda subcapa que forma la capa de sistema se compone principalmente de dos tipos de operaciones:

1. Demultiplexión.

[2]Esta es la situación que vamos a tener durante la generación de los subtítulos.

2. Sincronización de la reproducción de múltiples flujos elementales.

3.2.5.1. Demultiplexión

Durante la codificación, los flujos elementales se dividen temporalmente en paquetes que son serializados. Estos paquetes constituyen a su vez unidades lógicas llamadas paquetes PES cuya información procede de un único flujo elemental.

En cambio, durante la decodificación, es necesario deshacer dicho proceso, de forma que se reconstruya el flujo elemental original a partir del flujo de transporte o de programa multiplexado. Esta operación resulta bastante sencilla pues todos los paquetes van marcados con el *Stream_id* en el caso de los PS o con el *Packet_ID* en el caso de los TS, lo que permite identificar en todo momento los paquetes que pertenecen a cada flujo.

3.2.5.2. Sincronización

En primer lugar es importante resaltar los dos tipos de sincronización que tienen lugar durante la decodificación. Por un lado, es necesario realizar la **sincronización entre los diferentes flujos elementales que forman un programa**, es decir, los fragmentos de audio se deben corresponder con los de vídeo y el resto de flujos del programa (Subtítulos, etc.). Por otro lado, los decodificadores tienen que **sincronizarse con el canal**.

La sincronización entre los diferentes flujos que conforman un programa se consigue mediante los PTS, que tal y como ya vimos, presentan incrementos de 90 KHz. Durante la decodificación de los N flujos elementales, éstos se sincronizan ajustando la decodificación de cada flujo elemental a un base de tiempos común, cuyo origen puede ser el reloj de alguno de los N decodificadores, el reloj de la fuente de datos o algún reloj externo.

Hay que tener en cuenta que cada programa en un flujo de transporte puede tener su propia base de tiempos, y que las bases de tiempos de cada uno de los programas que forman un TS pueden ser diferentes entre sí. Dado que los PTS son aplicables a cada flujo en particular, estos se incluyen en la capa de paquetes PES tanto en los flujos de transporte como en los de programa.

Respecto a la sincronización del sistema de decodificado con el canal, ésta se consigue haciendo uso del SCR, en los flujos de programa, o del PCR, en el caso de los flujos de transporte. Ambos son marcas de tiempo que codifican la temporización del flujo de bits y que se obtienen de la misma base de tiempos que los PTS del audio y del vídeo correspondientes al mismo programa. Dado que cada programa puede tener su propia base de tiempos, existen PCRs diferentes para cada programa contenido en un flujo de transporte con múltiples programas. En algunos casos es posible que los programas compartan el valor de un PCR, pero en cualquier caso,

un programa solo va a poder tener un PCR asociado. Todo lo relacionado con el PCR se explicará mejor en la subsección 3.2.8.

3.2.6. Aplicaciones

Por las características que hemos definido hasta ahora, los dos tipos de flujos definidos, son especialmente adecuados dependiendo del propósito de la aplicación. Así, los flujos de programa son apropiados para la transmisión de datos en redes de comunicaciones modernas o para el almacenamiento de aplicaciones multimedia en CD-ROM.

En cambio, los flujos de transporte son adecuados para medios propensos a generar errores como redes de larga distancia o sistemas de emisión.

3.2.7. Estructura del Transport Stream

Recapitulando toda la información expuesta hasta ahora, hemos visto cómo un flujo de transporte permite combinar uno o más programas en un único flujo. Para ello, los datos de cada flujo elemental son multiplexados junto con información que nos permita sincronizar la presentación de los flujos elementales pertenecientes a un programa.

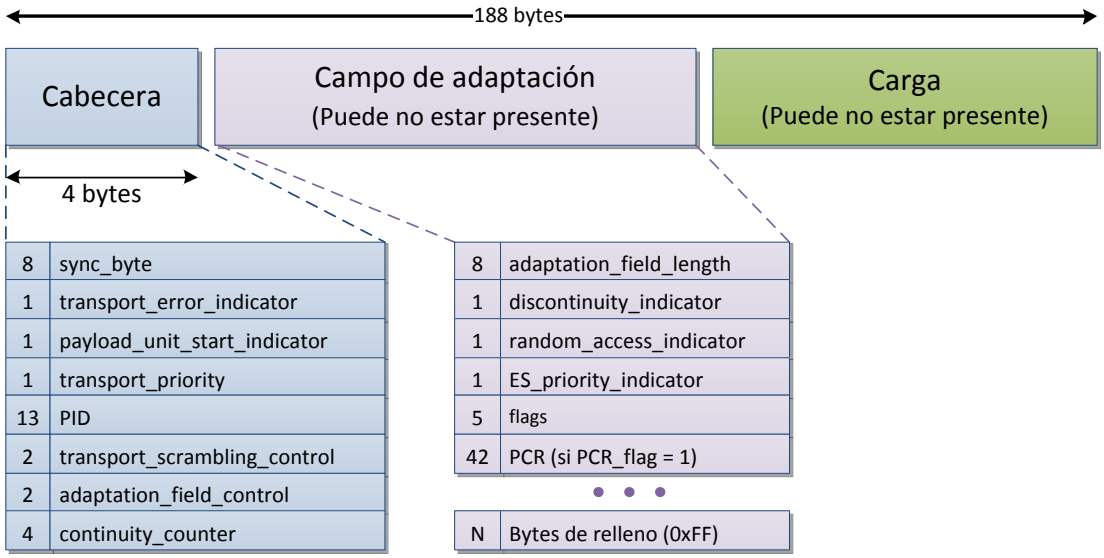


Figura 3.5: Estructura de un paquete de transporte.

Por otro lado, los datos de cada flujo elemental se transportan en paquetes PES. Estos paquetes PES, cuya definición se realizó en la subsección 3.2.3, son insertados en paquetes de transporte. El primer byte de la cabecera de cada paquete PES

debe estar situado en la primera posición disponible de la carga de un paquete de transporte.

Asimismo, definimos los paquetes de transporte como paquetes de 188 bytes en los cuales, los primeros 4 bytes corresponden a la cabecera. La Figura 3.5 en la página anterior muestra la estructura de un paquete de transporte en el que se aprecian tres campos bien diferenciados: la cabecera, el campo de adaptación, y por último, la carga del paquete. A continuación explicamos cada uno de estos campos.

3.2.7.1. Cabecera del paquete de transporte

En la Figura 3.6 se puede observar un diagrama de bloques que determina la posición y tamaño de los campos de la cabecera del paquete de transporte. El significado de los campos es el siguiente:

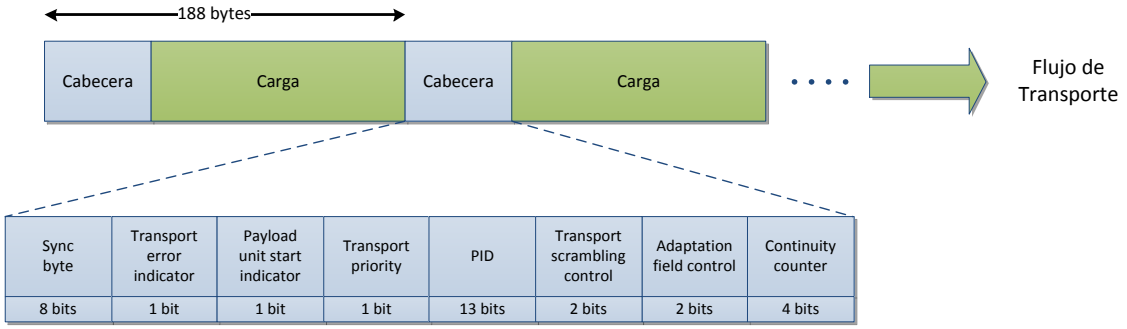


Figura 3.6: Cabecera de un paquete de transporte [14].

1. *sync_byte*: el byte de sincronización es un campo fijo cuyo valor en binario es “0b01000111” (En hexadecimal, “0x47”).

2. *transport_error_indicator*: *flag* de 1 bit. Cuando se pone a "1" indica que al menos 1 error incorregible existe en el paquete. Este flag puede ser puesto de forma externa a la capa de transporte y no debe ser puesto a "0" hasta que el error sea corregido.

3. *payload_unit_start_indicator*: flag de 1 bit. Tiene significado cuando los paquetes de transporte llevan datos de un paquete PES o datos *Program Specific Information* (PSI). Si su valor es "1", significa que la carga de este paquete comienza con el primer byte de un paquete PES o de una sección PSI. En cambio, si su valor es "0", indica que no hay inicio de paquete PES o sección PSI en este paquete de transporte. Para los paquetes inválidos su valor es "0".

4. *transport_priority*: es un indicador de 1 bit. Cuando es fijado a "1" indica que el paquete es de mayor prioridad que otros paquetes con el mismo PID que no tienen este bit puesto a "1".

5. PID: es un campo de 13 bits cuya misión es identificar, a través de las tablas PSI, el tipo de los datos almacenados en la carga del paquete. La Tabla A.1, contenida en el anexo A, refleja los posibles valores que puede adoptar el PID[3].

6. *transport_scrambling_control*: este campo de 2 bits indica si la carga del paquete ha sufrido *scrambling*, es decir, ha sido desordenada. Si su valor es "00", indica que no se ha producido scrambling, en cualquier otro caso, indica lo contrario. Es importante tener en cuenta que ni la cabecera ni el campo de adaptación pueden ser desordenados.

7. *adaptation_field_control*: campo de dos bits que especifica si a la cabecera le sigue un campo de adaptación o la carga del paquete. Estos son sus posibles valores:

   a) "00": reservado para uso futuro.

   b) "01": Únicamente carga.

   c) "10": Sólo campo de adaptación.

   d) "11": Campo de adaptación seguido de la carga.

8. *continuity_counter*: es un contador circular de 4 bits cuyo valor se incrementa en cada paquete de transporte con el mismo PID. Cuando el contador llega a su valor máximo, esto es 15, regresa a 0 en el siguiente paquete. El valor del contador no se debe incrementar en aquellos casos en los que el adaptation_field_control sea igual a "00" o "10".

3.2.7.2. Campo de adaptación

Este campo tiene principalmente dos funciones:

1. Codificar el PCR del flujo de entrada.

2. Completar el último paquete con bytes de relleno en aquellos casos en los que el tamaño de un paquete PES no de para llenar un múltiplo entero de paquetes de transporte.

A continuación se detallan algunos de los campos que forman el campo de adaptación[4], cuya estructura se ve reflejada en la Figura 3.7 en la página siguiente. Como se puede apreciar, no solo es un campo variable en tamaño, sino también en contenido. Dependiendo de la función que tenga que desempeñar, presentará unos campos u

[3]Los paquetes de transporte con valores de PID 0x0000, 0x0001, y 0x0010 - 0x1FFE pueden llevar un PCR.

[4]Durante el desarrollo del proyecto se utiliza este campo en su forma más básica: Tamaño del campo de adaptación + flags (8bits) + bytes de relleno.

otros, los cuales serán advertidos por el valor de un conjunto de flags. Dichos flags están identificados en la Figura 3.7 y son aquellos de los que parte una flecha hacia los campos opcionales que habilita.

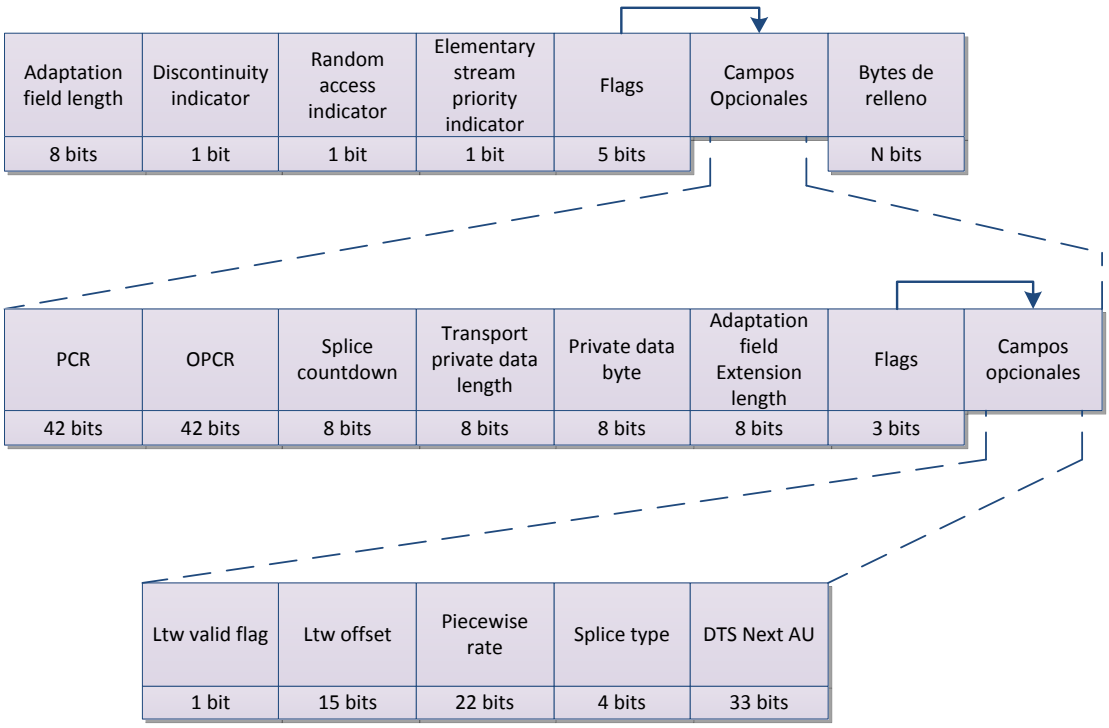


Figura 3.7: Campo de adaptación de un paquete de transporte [14].

1. *adaptation_field_length*: campo de 1byte que especifica el número de bytes del campo de adaptación justo a partir del adaptation_field_length. Para insertar un sólo byte de relleno, el valor de este campo se fija a 0, consiguiendo el relleno a través de los 8 bits siguientes que corresponden a flags.

2. *PCR_flag*: flag de 1 bit. Si su valor es "1" indica que el campo de adaptación contiene un campo PCR codificado en 2 partes dentro de la parte opcional del campo de adaptación. Un "0" indica lo contrario.

3. *adaptation_field_extension_flag*: indicador de 1 bit que advierte de la presencia de una extensión del campo de adaptación cuando su valor es "1". Si observamos la Figura 3.7, la tercera fila de bloques aparecerá cuando este flag valga "1".

4. *PCR*: tal y como mencionamos en 3.2.2, el PCR es un campo de 42 bits que indica el tiempo supuesto de llegada del byte que contiene el último bit del campo program_clock_reference_base.

5. *adaptation_field_extension_length*: campo de 8 bits que indica el número de bytes de la extensión del campo de adaptación que se encuentra inmediatamente a continuación de este campo.

6. *stuffing_bytes*: constituyen los bytes de relleno del campo de adaptación. Son introducidos por el codificador con el valor "0xFF" y descartados durante la decodificación.

3.2.7.3. Generación de un flujo de transporte

Una vez entendido el significado de un paquete PES y de un paquete de transporte, estamos en posición de entender cómo se genera un flujo de transporte a partir de un flujo elemental sea del tipo que sea. En la Figura 3.8 se explica este proceso tomando como fuente de datos un flujo elemental de vídeo. No obstante, esto es extensible a la generación de un flujo a partir de varios flujos elementales.

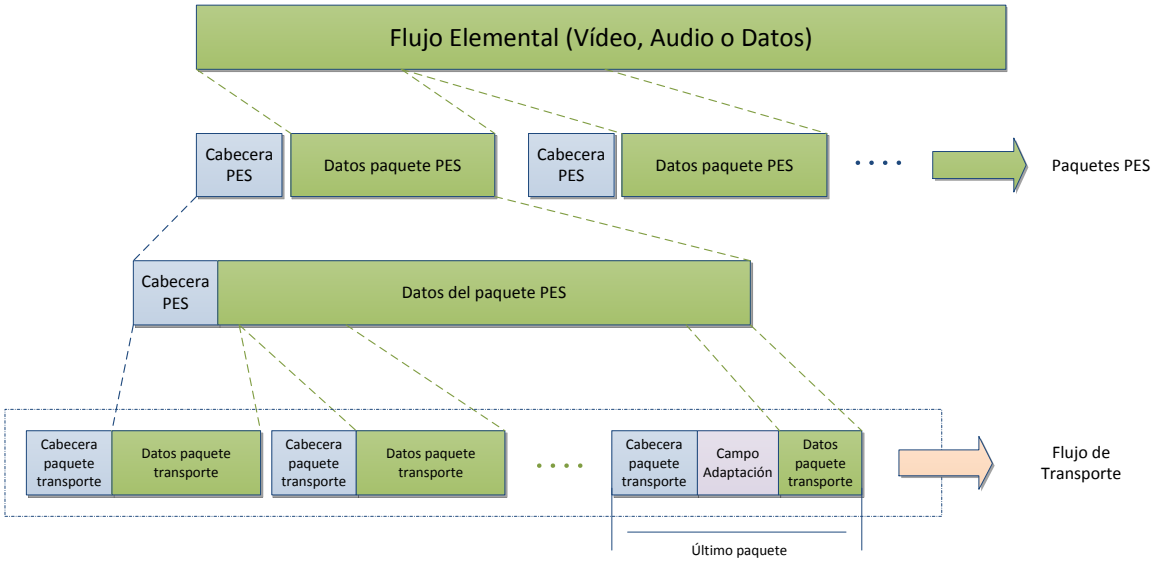


Figura 3.8: Generación de un flujo de transporte [14].

Si nos fijamos en el gráfico, en primer lugar se divide la información contenida en el flujo elemental de vídeo en paquetes PES, que en este caso contendrán los diferentes frames que forman el vídeo. Por lo general, estos paquetes van a ser mucho más grandes que los paquetes de transporte por lo que es necesario dividir el paquete PES en paquetes de transporte, tal y como se observa en la parte inferior de la figura. Además, en el caso de que el último paquete no quede relleno al completo por los datos del paquete PES, será necesario introducir un campo de adaptación tal y como se puede contemplar en el gráfico.

Todos estos paquetes forman el flujo de transporte. En caso de tener más flujos elementales, se repetiría el mismo proceso para cada uno de ellos, lo que daría lugar

a más paquetes de transporte que viajarían mezclados unos con otros. Esto no va a representar ningún problema para el decodificador, pues como ya hemos visto, el flujo de transporte está dotado de suficiente información para que el decodificador pueda reconstruir los flujos elementales y presentarlos de forma sincronizada.

3.2.8. *Program Specific Information*

Es la información usada por el decodificador para conocer cómo es el flujo de transporte de entrada. Los datos PSI proporcionan la información necesaria para que se pueda realizar la configuración automática del receptor a la hora de demultiplexar y decodificar los diferentes flujos de los programas que forman el múltiplex.

En los flujos de transporte, la **PSI se clasifica en cinco tablas** tal y como podemos observar en la Tabla 3.1 en la página siguiente[5] [6]. La norma MPEG-2 Systems propone que las tablas PSI deben ser segmentadas en secciones que serán transportadas dentro de paquetes de transporte cuyos PIDs serán predeterminados en algunos casos, y seleccionables por el usuario en otros (ver anexo A). De esta forma el decodificador es capaz de ir reconstruyendo las tablas que describen al flujo de transporte conforme vaya leyendo los datos del flujo correspondientes a las secciones[7] (en la subsección 3.2.8.5 se describe gráficamente este proceso).

Las secciones pueden ser de longitud variable. El inicio de una sección es indicado por el campo denominado como *pointer_field*, que siempre se encuentra en el primer byte de la carga de un paquete de transporte. El pointer_field es un campo de 8 bits cuyo valor representa el número de bytes que hay entre él y el primer byte de la primera sección presente en la carga del paquete. En el caso de que al menos una sección empiece en un paquete de transporte, entonces el flag payload_unit_start_indicator debe ser puesto a "1". En cambio, **si ninguna sección empieza** en este paquete, este flag se pone a "0" y se omite el envío del puntero.

Una vez interpretado el pointer_field, el decodificador va a ser capaz de construir las tablas PSI a partir de las secciones que se exponen a continuación.

[5] La NIT no se explica en esta norma sino que es expuesta en ETSI EN 300 468. A pesar de que según la norma, el número de PID de la NIT es asignado en la PAT, según la definición de ETSI EN 300 468, su valor es 0x10.

[6] *Entitlement management message* (EMM - ver Tabla 3.1): es información privada de acceso condicional que especifica los niveles de autorización o los servicios de decodificadores específicos. Esta tabla no resulta relevante teniendo en cuenta los objetivos de este proyecto, por lo que únicamente se hará una breve descripción de ella.

[7] Esta es la razón por la que cuando un usuario selecciona un canal de la TDT, no se produce la visualización instantánea de la imagen y el audio, ya que el receptor necesita capturar los paquetes en los que se encuentran las secciones que forman la PSI y, de esta manera, conocer qué paquetes son los que contienen la información de vídeo, audio, subtítulos, etc.

Nombre Tabla	Número de PID	Descripción
Program Association Table (PAT)	0x00	Asocia números de programa con el PID de su Program Map Table
Program Map Table (PMT)	Asignado en la PAT	Especifica el PID de los componentes que forman uno o más programas
Network Information Table (NIT)	Asignado en la PAT	Parámetros de red físicos tales como las frecuencias FDM, número de los transpondedores, etc.
Conditional Access Table (CAT)	0x01	Asocia uno o más flujos EMM con un único PID
Transport Stream Description Table (TSDT)	0x02	Asocia uno o más descriptores a un flujo de transporte

Tabla 3.1: Program specific information.

3.2.8.1. Program Association Table - PAT

Cada flujo de transporte debe contener uno o más paquetes de transporte cuyo PID es "0x0000". La información contenida en ellos permite reconstruir por completo la PAT, la cual contiene la lista de todos los programas que se encuentran en el flujo de transporte. Cada uno de estos programas es identificado mediante una etiqueta numérica denominada *program_number* a la que la PAT asocia el PID (*program_map_PID)* de los paquetes que contienen la definición del programa, es decir, los paquetes en los que se encuentra la sección con la que formar la tabla PMT y que describiremos en la sección siguiente.

Existe una salvedad para el número de programa 0. Cuando aparece un número de programa con este valor en la PAT, siempre se refiere a la NIT (ver sección 3.3). En ese caso, el PID asociado no es el de la PMT sino el de los paquetes con la sección de la NIT.

La Figura 3.9 muestra la estructura de la sección *program_association_section*, la cual comparte algunos campos con los otros tipos de secciones. De entre estos campos nos vamos a centrar en dos ellos cuyo significado va a ser el mismo para las secciones de los apartados siguientes.

- *table_id*: campo de 8 bits que identifica los contenidos de las secciones PSI de un flujo de transporte de acuerdo a los valores expuestos en la Tabla B.1 del anexo B.

- *current_next_indicator*: flag de 1 bit que, cuando vale "1", indica que la PAT descrita por estos paquetes es aplicable de manera inmediata (igual para la

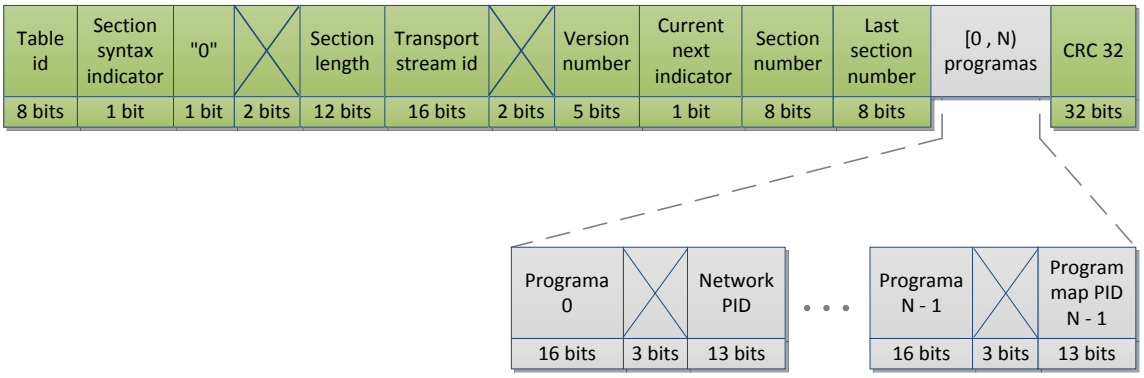


Figura 3.9: Program association table (PAT) [14].

PMT, la CAT y la TSDT). En caso de valer "0", significa que la tabla enviada todavía no es aplicable pero que va a ser la siguiente en ser válida.

El tamaño de esta sección dependerá de la cantidad de programas contenidos en el flujo de transporte. De ahí que los campos sombreados puedan variar entre 0 y N - 1 programas.

3.2.8.2. Program Map Table - PMT

La Program Map Table es la tabla en la que se almacena la información que relaciona los números de programa descritos por la PAT con los elementos que forman dicho programa, entre los que se encuentran los flujos elementales de vídeo, audio y subtítulos. Cada asociación entre un número de programa y los elementos que lo conforman recibe el nombre de definición de programa, por lo que la PMT es la recopilación de todas las definiciones de programa de un flujo de transporte.

Un flujo de transporte debe contener uno o más paquetes de transporte, cuyos valores de PID sean iguales a los asociados en la PAT a sus respectivos números de programa. Todos los programas contemplados en la PAT deben ser descritos un una sola sección, denominada como *TS_program_map_section*.

Los datos privados cuyo PID aparece en la PMT como *elementary_PID* pertenecen igualmente a ese programa. En cambio, puede haber datos privados dentro del flujo de transporte que no pertenezcan a ningún programa, por lo que no aparecerán listados en la PMT. Al igual que se explicó en la PAT, la versión más reciente de una sección de la PMT será aquella cuyo current_next_indicator valga “1”. **Durante el tiempo de existencia de un programa, su program_map_PID no debe cambiar**.

La Figura 3.10 representa la estructura de la sección TS_program_map_section de la que vamos a describir sus campos más relevantes.

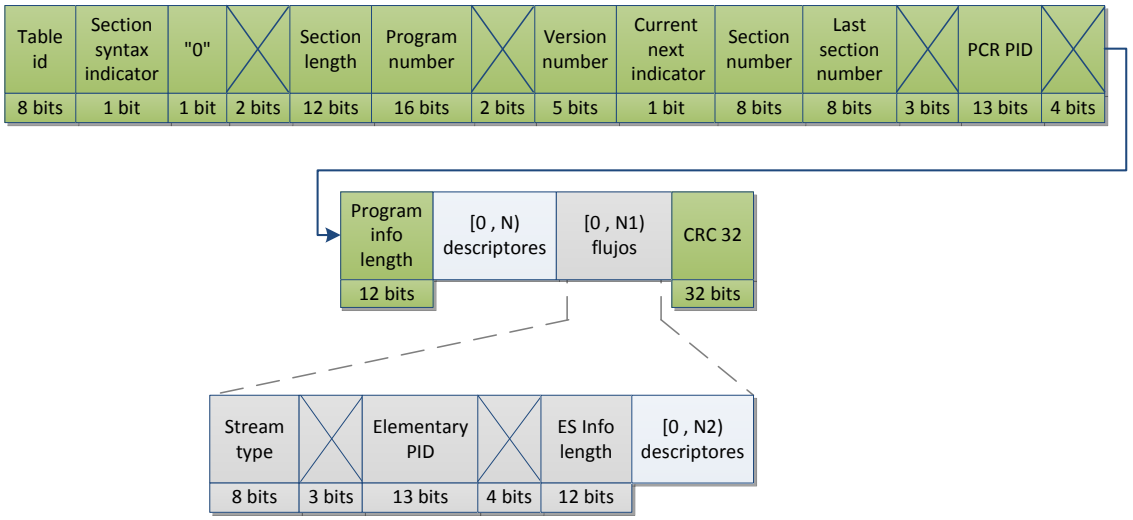


Figura 3.10: Program map table (PMT) [14].

- *table_id*: para la PMT su valor será "0x02" tal y como se refleja en la Tabla B.1.
- *program_number*: es el número de programa con el que está asociado el program_map_PID en la PAT.
- *PCR_PID*: campo de 13 bits que indica el PID de los paquetes de transporte que van a contener los PCRs validos para el programa apuntado por program_number. Si no hay ningún PCR asociado para los flujos privados, entonces toma el valor "0x1FFF".
- *stream_type*: es el campo que especifica el tipo de elemento perteneciente al programa y que va a ir contenido en los paquetes de transporte cuyo PID es *elementary_PID*.
- descriptores: son estructuras usadas para extender la definición de los programas y de sus elementos. Todos los descriptores comienzan con un tag de 8 bits seguido de un campo, también de 8 bits, en el que se guarda el tamaño del descriptor. Dentro de esta norma se describen los descriptores necesarios para completar la información PSI. Sin embargo, no son los únicos que existen, ya que en ETSI EN 300 468 se describen los descriptores correspondientes a la información que se va a definir en la sección 3.3 como SI.

3.2.8.3. Conditional Access Table - CAT

La tabla de acceso condicional proporciona la asociación existente entre uno o más sistemas de acceso condicional, sus flujos EMM y cualquier parámetro especial

asociado a ellos.

Este tipo de tablas serán enviadas en el flujo de transporte toda vez que uno o más flujos elementales del flujo de transporte hayan sufrido scrambling. El PID de los paquetes de transporte encargados de llevar la información de la CAT debe ser "0x0001" y todos juntos, deben formar una versión completa de la tabla de acceso condicional cuya estructura es descrita en la Figura 3.11. En cuanto al table_id de esta sección, su valor será "0x01".

Table id	Section syntax indicator	"0"		Section length		Version number	Current next indicator	Section number	Last section number	[0 , N) descriptores	CRC 32
8 bits	1 bit	1 bit	2 bits	12 bits	18 bits	5 bits	1 bit	8 bits	8 bits		32 bits

Figura 3.11: Conditional access Table (CAT) [14].

3.2.8.4. Transport Stream Description Table - TSDT

La Transport Stream Description Table es opcional y está definida para realizar el transporte de los descriptores empleados para extender la definición de los programas, los cuales ya fueron descritos en la subsección 3.2.8.2. El table_id que corresponde a esta tabla es el "0x03" la cual es transportada en paquetes cuyo PID debe ser el "0x0002".

La estructura de esta tabla es exactamente igual a la de la CAT por lo que no es necesario incluir ninguna figura que la describa.

3.2.8.5. Proceso de reconstrucción de las tablas PSI

Para concluir la descripción de este estándar, se detalla mediante la Figura 3.12 en la página siguiente, una simplificación del modo en que la información PSI viaja en un flujo de transporte junto a los flujos elementales que constituyen un programa, así como la forma en que los paquetes se relacionan unos con otros. A modo de ejemplo, se procede a explicar los pasos que seguiría un decodificador para visualizar el contenido de un canal en concreto[8]:

1. En primer lugar, el decodificador debe leer los paquetes de transporte cuyo PID sea el 0, es decir, los paquetes que llevan la PAT y construir la versión de la PAT más reciente.

2. Dentro de la PAT, buscar el PID de la PMT del número de programa que se quiere decodificar. Para el programa 530 el PID a buscar sería el 100.

[8]Los valores que se muestran en la figura son reales, y fueron extraídos mediante un analizador de flujos de transporte.

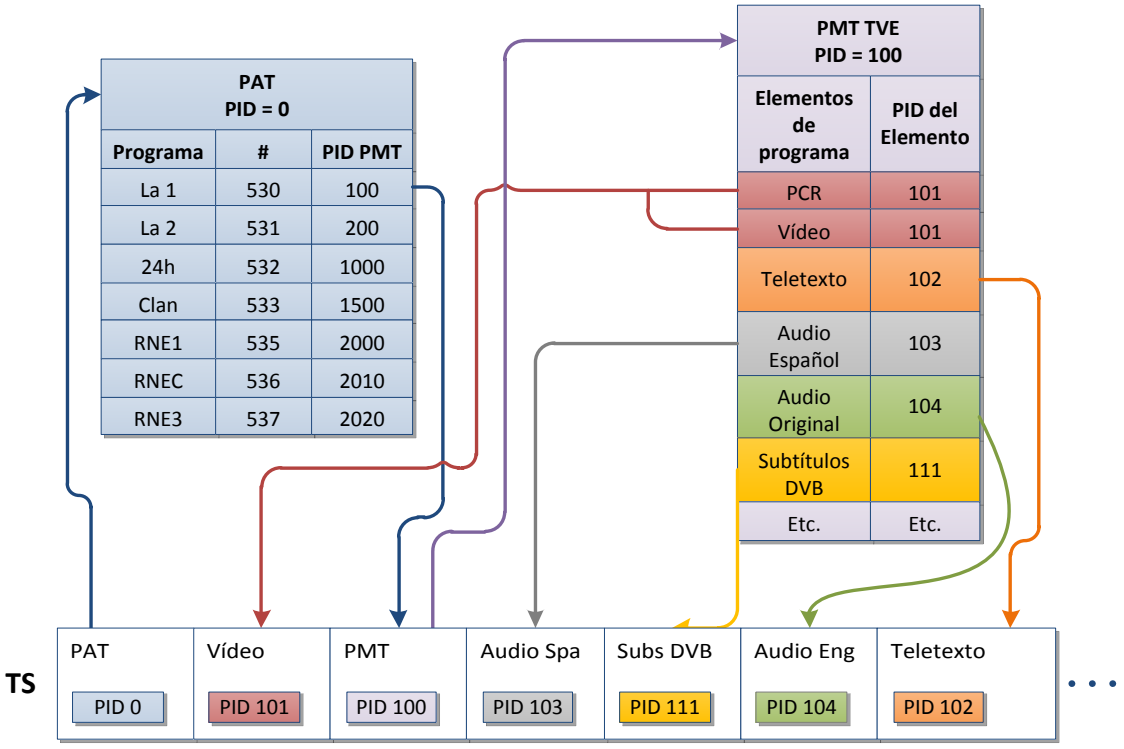


Figura 3.12: Proceso de reconstrucción de las tablas PSI.

3. Leer los paquetes cuyo PID sea 100 para construir la PMT completa.

4. Decodificar los flujos elementales del programa a partir de los PIDs indicados por la PMT:

   a) Decodificación del vídeo y del PCR: tal y como se puede observar, el PCR va incluido en los paquetes que transportan el flujo elemental de vídeo. La decisión de incluirlo en estos paquetes es tomada por las cadenas de televisión ya que este aspecto se deja abierto en el estándar[9]. Para poder presentar el vídeo, el decodificador debe encontrar los paquetes cuyo PID sea el 101. De esta manera, será capaz de regenerar los paquetes PES que forman el flujo elemental de vídeo.

   b) Decodificación del audio: para poder presentar el audio que se corresponde con la imagen en pantalla, el usuario debe indicar al sistema receptor, qué tipo de audio desea escuchar. En caso de seleccionar el idioma castellano, según el ejemplo, los paquetes a leer son aquellos cuyo PID es el 103.

[9]Dado que una cadena de televisión siempre va a transmitir un flujo elemental de vídeo, la decisión de incluir el PCR en estos paquetes resulta la más acertada. De igual manera, las cadenas de radio incluyen los paquetes de PCR en el flujo elemental de audio.

c) Decodificación de los subtítulos: al igual que el audio, el usuario tiene dos formas de visualizar los subtítulos. En caso de desearlos en formato DVB, el objetivo del decodificador va a ser encontrar los paquetes con PID 111, mientras que si se desean en formato teletexto, el decodificador deberá localizar los paquetes con PID 102, ya que este tipo de subtítulos van contenidos en una página del teletextoEn caso de no desear subtítulos el decodificador obviará estos paquetes.

3.3. Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems. ETSI EN 300 468

La norma referenciada como ETSI EN 300 468 aporta información adicional a la definida como PSI, proporcionando datos de ayuda durante la sintonización automática de los IRDs, así como información complementaria para ser presentada al usuario. La forma en la que se debe presentar dicha información no se especifica en la norma por lo que este aspecto se deja en manos de los fabricantes de IRDs [15].

En la sección anterior, se explicó cómo la información denominada como PSI se estructuraba en cinco tablas: PAT, CAT, PMT, TSDT y NIT. No obstante, a pesar de que la NIT pertenece a este grupo de tablas, es definida en este documento en conformidad con la especificación ISO/IEC 13818-1 [14], siendo su principal objetivo, proveer información acerca de la red física.

A diferencia de la PAT, la CAT y la PMT, las cuales proporcionan únicamente información del múltiplex en el que están contenidas, **la información de servicio definida en este documento puede proporcionar también información sobre los servicios y eventos transportados por otros múltiplex e incluso sobre otras redes**. La acción conjunta de ambos tipos de información de servicio es lo que permite hacer uso de MPEG para transportar la televisión digital.

La Figura 3.13 en la página siguiente describe el mapa completo de la información de servicio que viaja en un flujo de transporte. En su parte izquierda, se puede observar la PSI descrita en la sección anterior, mientras que en la derecha, las tablas que describe la norma en cuestión. En un primer lugar, y para terminar con la información definida como PSI, vamos a explicar la NIT, para a continuación centrarnos en la información de servicio, la cual se estructura en las siguientes nueve tablas: BAT, SDT, EIT, RST, TDT, TOT, ST, SIT y DIT.

Por último, de entre todos los descriptores que se citan en la norma, nos vamos a centrar en uno de ellos dada la importancia que tiene en relación con el proyecto: el *Subtitling Descriptor* (Descriptor de subtitulado).

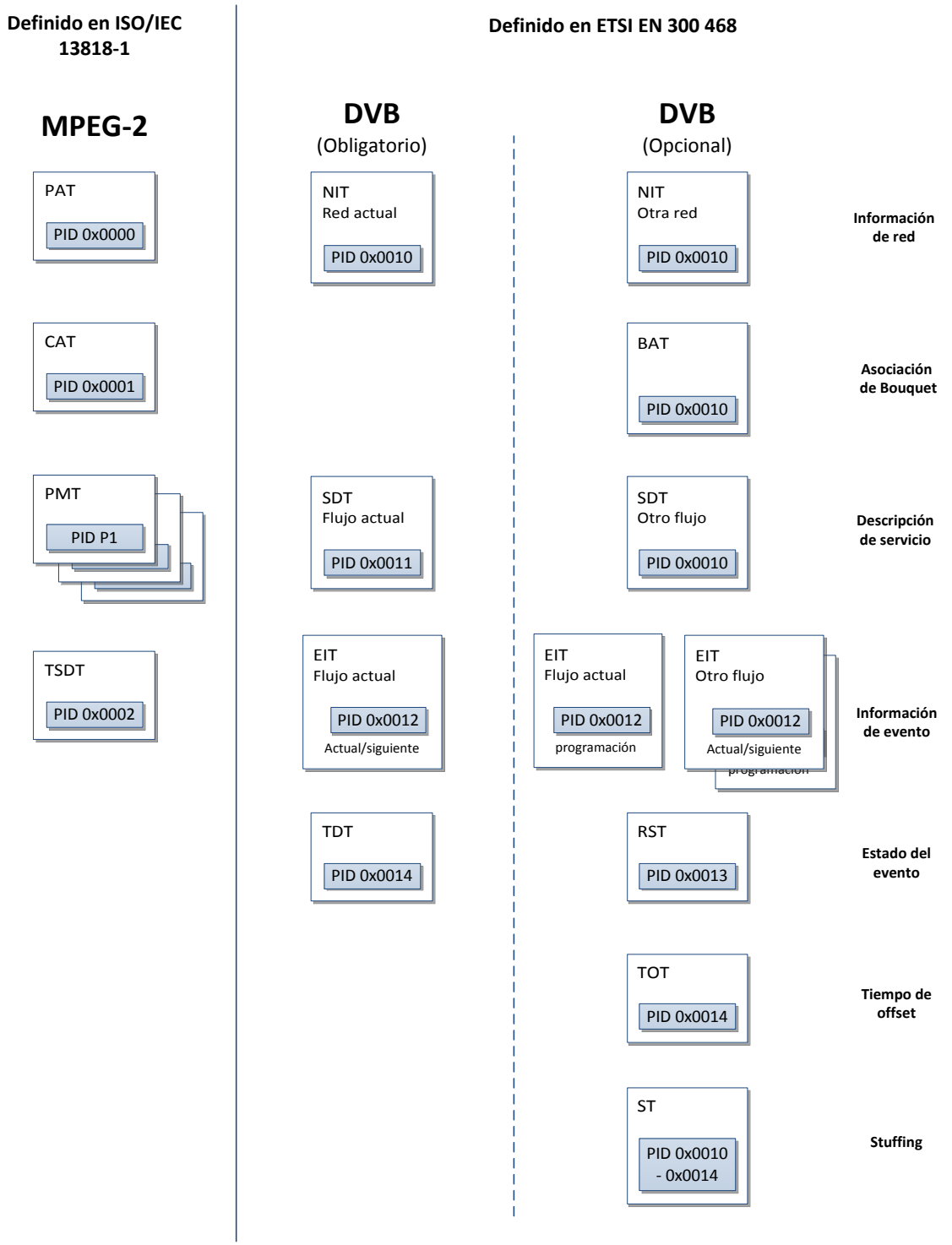


Figura 3.13: Organización general de la información de servicio (SI) [15].

3.3.1. Network Information Table - NIT

La NIT transporta información relativa a la organización física de los flujos de transporte que son transmitidos a través de una red, así como las características de la propia red. Cada una de las redes recibe un *network_id* individual que sirve para identificar de forma unívoca a las mismas.

En algunos casos, los receptores son capaces de almacenar la información de la NIT en memoria no volátil de tal forma que se minimice el tiempo de acceso durante el cambio entre canales.

Las secciones encargadas de transmitir la información necesaria para construir la NIT deben ser transmitidas en paquetes de transporte cuyo PID sea "0x0010". Además, las secciones de la NIT que describen la red actual deben presentar un table_id de valor "0x40" mientras que si se refiere a otra red este campo valdrá "0x41".

3.3.2. Bouquet Association Table - BAT

La BAT proporciona información sobre lo que se define como *bouquets*. Además de identificarlos mediante el *bouquet_id*, la BAT incluye la lista de servicios que conforman el bouquet. Como se puede observar en la Figura 3.14, un bouquet se compone de servicios que pueden proceder del mismo multiplex, de diferentes multiplex o incluso de servicios procedentes de otras redes.

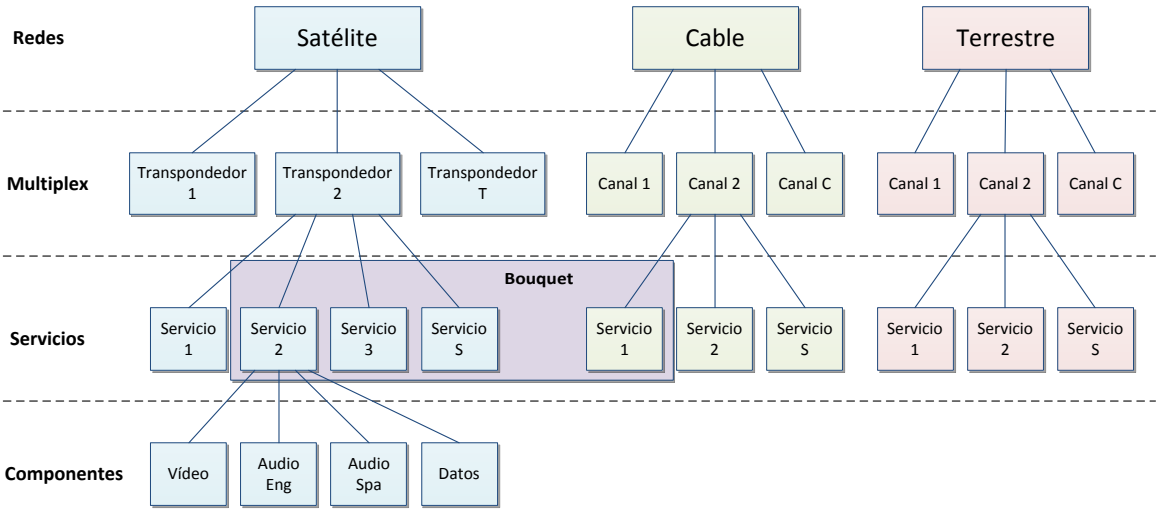


Figura 3.14: Modelo de entrega de servicios [15].

Las secciones que forman parte de la BAT deben ser transmitidas en paquetes de transporte con valor de PID "0x0011" y cuyo table_id sea "0x4A".

3.3.3. Service Description Table - SDT

La SDT contiene información para describir los servicios contenidos en un flujo en particular (nombre del servicio, proveedor del servicio, etc.). Los servicios pueden ser parte del flujo de transporte actual o parte de otro flujo. Las secciones que describen la SDT deben ser transmitidas en paquetes de transporte identificados con el PID "0x0011". En caso de que las secciones describan el flujo sobre el que viajan, deben presentar como table_id el valor "0x42". En cambio, si se refieren a otro flujo, entonces este campo valdrá "0x46".

3.3.4. Event Information Table - EIT

La EIT proporciona información en forma cronológica concerniente a los eventos que componen un servicio (nombre del evento, hora de inicio, duración, etc.). Se han definido cuatro tipos de información de eventos para los diferentes tipos de servicios, cada uno de ellos con diferentes valores de table_id:

1. Flujo de transporte actual. Información del evento actual y del siguiente. table_id = "0x4E".

2. Flujo de transporte diferente al actual. Información del evento actual y del siguiente. table_id = "0x4F".

3. Flujo de transporte actual. Información sobre la programación de eventos. table_id = "0x50" - "0x5F".

4. Flujo de transporte diferente al actual. Información sobre la programación de eventos. table_id = "0x60" - "0x6F".

Las tablas de los puntos 1 y 2 contienen información perteneciente al evento actual y al siguiente en orden cronológico, los cuales son proporcionados por un servicio dado en el flujo de transporte actual o en otro diferente, excepto en el caso de un servicio de *Near Video On Demand* (NVOD), el cual puede tener más de dos descripciones de eventos.

En cuanto a las tablas de los puntos 3 y 4, contienen una lista de eventos en forma de parrilla de programación que se puede referir al flujo actual o a otro. Estas tablas son opcionales y su información debe estar ordenada cronológicamente.

Las secciones que forman parte de la EIT deben ser transmitidas en paquetes de transporte cuyo PID es el "0x0012".

3.3.5. Running Status Table - RST

La tabla RST permite actualizar de forma rápida y precisa el estado temporal de uno o más eventos. Su razón de ser radica en los desajustes producidos por los

cambios en la planificación, para los que tener una tabla separada con información actualizada de la planificación permite conseguir el reajuste de la manera más rápida posible.

Los paquetes que transportan las secciones de la RST tienen el PID "0x0013" y table_id "0x71".

3.3.6. Time and Date Table - TDT

La TDT es una tabla muy sencilla que transporta únicamente la hora y fecha UTC[10]. Dado que esta información es actualizada frecuentemente, se transporta en una tabla separada para que sea fácilmente actualizable.

Esta tabla se transporta en una sola sección cuyo PID será "0x0014". El table_id tendrá siempre el valor "0x70".

3.3.7. Time Offset Table - TOT

La TOT es una tabla muy parecida a la TDT con la diferencia de que no sólo contiene la hora y fecha UTC, sino que también presenta el desfase respecto a la hora local. Al igual que la TDT, se transporta en una tabla diferente para facilitar su actualización. Los paquetes de la TOT van marcados con el PID "0x0014" y table_id "0x73".

3.3.8. Stuffing Table - ST

La tabla ST se usa para invalidar las secciones existentes en los extremos de un sistema de transmisión. Esta circunstancia se da cuando una tabla es sobrescrita y todas las secciones que la forman deben ser sobrescritas para mantener la integridad de la tabla.

3.3.9. Selection Information Table (SIT) y Discontinuity Information Table (DIT)

Para las dos últimas tablas que quedan por describir, es necesario hacer una breve descripción previa sobre lo que se conoce como flujo de transporte incompleto. Este tipo de flujos no son conformes a las especificaciones normales y contienen un subjuego de los flujos que forman el flujo de transporte original. Además, pueden ser discontinuos, es decir, puede haber cambios en el flujo parcial y existir discontinuidades temporales. Con el fin de transportar la información de servicio del flujo parcial, surgen estas dos tablas.

[10]Tiempo universal coordinado. Es el tiempo de la zona horaria de referencia respecto a la cual se calculan todas las otras zonas del mundo.

3.3.9.1. Selection Information Table

Por su parte, la SIT porta un resumen de la información SI requerida para describir los flujos elementales del flujo de transporte parcial.

3.3.9.2. Discontinuity Information Table

La tabla DIT tiene el propósito de ser insertada en los puntos de transición en los que la información SI puede ser discontinua.

3.3.10. Subtitling Descriptor

EL descriptor de subtitulado se encuentra estrechamente ligado a la PMT, ya que entre otras cosas, es la única posible localización donde se puede encontrar. El valor de stream_type que se asocia a los subtítulos y por consiguiente, a todos los paquetes cuyo PID sea el definido por el campo *Elementary_PID* de la PMT que hace referencia al flujo de subtítulos, es el "0x06".

Durante la explicación que se hizo de los descriptores en la subsección 3.2.8.2, mencionamos que todo descriptor comienza con dos campos de 8 bits: el descriptor_tag y el descriptor_length. En el caso del descriptor de subtitulado, el descriptor_tag debe valer "0x59" mientras que la longitud del descriptor variará en función de los servicios de subtitulado que formen el flujo de subtítulos. La Figura 3.15 muestra en detalle la estructura y composición del descriptor, cuyos campos son los siguientes:

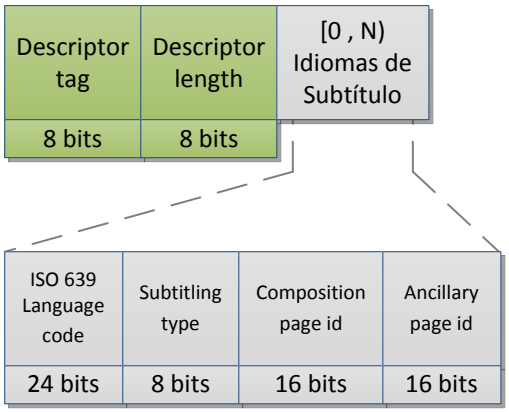


Figura 3.15: Descriptor de Subtitulado.

- *ISO_639_language_code*: Campo de 24 bits que contiene el código de lenguaje de tres caracteres del subtítulo definido en ISO 639-2. Cada carácter se codifica

en 8 bits y es insertado en orden. Para el caso de que el subtítulo esté en español o inglés, el código a insertar sería “spa” o “eng” respectivamente.

- *subtitling_type*: este campo de 8 bits proporciona información sobre el contenido del subtítulo y la forma en que se va a mostrar al usuario. Los códigos posibles para este campo son los que se muestran en la Tabla C.1 del anexo C. Si nos fijamos en su contenido, se puede ver como existe una gran variedad de posibilidades, en especial para los subtítulos DVB, entre los que se hace distinción dependiendo del formato de emisión o del usuario al que va destinado.

- *composition_page_id*: Este campo de 16 bits identifica a la página de composición. Los segmentos de subtitulado DVB cuya *page_id* coincida con este campo, es decir, los segmentos que pertenecen a la página de composición, deben ser decodificados si los datos del descriptor de subtitulado coinciden con el criterio de selección realizado por el usuario. El identificador apuntado por este campo **debe aparecer al menos en los segmentos de subtitulado que definen la estructura de los datos en la página de subtitulado**: la *Page Composition Segment* (Segmento de composición de página) y la *Region Composition Segment* (Segmento de composición de región). Adicionalmente puede aparecer en el resto de tipos de segmentos.

- *ancillary_page_id*: identifica a la página auxiliar (*ancillary page)*, la cual es opcional. Al igual que los segmentos de la página de composición, los segmentos de la página auxiliar deben ser decodificados si los datos del descriptor de subtitulado coinciden con el criterio de selección realizado por el usuario. En caso de que no exista la página auxiliar, el valor de este campo será el mismo que el del campo *composition_page_id*. A diferencia del campo anterior, **este identificador nunca puede aparecer en los segmentos de composición**. Solo podrá aparecer en los segmentos que definen la CLUT, los objetos y otro tipo de segmentos.

Para comprender el significado y propósito de la página de composición y de la página auxiliar, es imprescindible comprender y entender los contenidos de la norma ETSI EN 300 743 [16], los cuales se detallan en la sección siguiente.

3.4. Digital Video Broadcasting (DVB); Subtitling Systems. ETSI EN 300 743

El objetivo de esta norma consiste en definir la sintaxis y procedimientos para decodificar los flujos de subtítulos. Un flujo de subtítulos se compone de uno o más

servicios de subtítulos, cada uno de los cuales, compuesto por la información textual y/o gráfica necesaria para proporcionar los subtítulos. El caso típico de uso de varios servicios de subtítulos en el mismo flujo es el de proporcionar subtitulado en varios idiomas [16].

Cada servicio de subtítulos muestra la información en una secuencia de páginas que se superponen a la imagen del vídeo asociado. Cada una de estas páginas puede contener una o más regiones rectangulares con unas características determinadas. Entre estas características están su tamaño horizontal y vertical, la resolución en píxeles o el color de fondo. A su vez, las regiones pueden contener objetos gráficos que deben estar contenidos dentro de los límites de la región y cuya posición queda definida en la *region composition segment* (RCS - Segmento de composición de región).

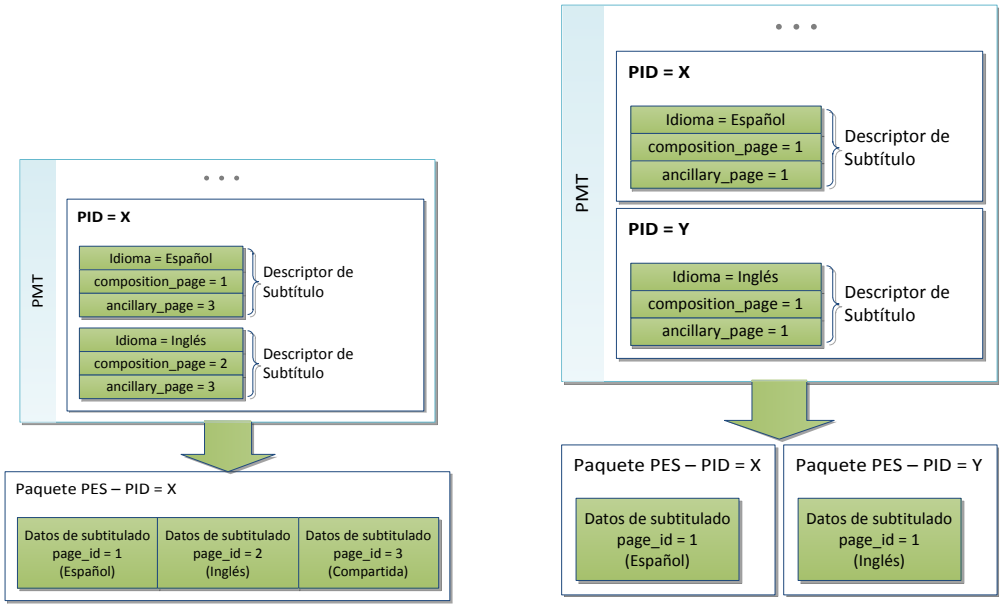
Por su parte, el uso y posicionamiento de las regiones dentro de una página se define en la *page composition segment* (PCS - Segmento de composición de página), en la cual, se proporciona una lista de regiones, cada una de ellas con su posición espacial. Es posible declarar regiones y que posteriormente no se usen, o usar más de una región a la vez dentro de una página. En cualquier caso, una página de composición no tiene por qué cambiar a pesar de que se añadan o quiten objetos, ya que esto sucede a nivel de región y no afecta a la página.

Al igual que los flujos de vídeo o de audio, el flujo de subtítulos DVB es transportado en paquetes PES y las marcas temporales relativas a su presentación, están definidas en el PTS de la cabecera PES. Este PTS determinará el instante en el que los contenidos de la página son mostrados en pantalla.

Para hacer un uso eficiente de la memoria del decodificador, el sistema de subtitulado DVB usa gráficos basados en regiones con colores de píxel indexados. Las resoluciones de píxel permitidas son 2, 4 y 8 bits lo que nos permite tener hasta 4, 16 o 256 códigos de píxel respectivamente. Cada una de las regiones se asocia con una, y sólo una CLUT, para definir los colores y transparencia de cada uno de los píxeles (La CLUT permite traducir el código del píxel a un color). En la mayoría de los casos, con una CLUT es suficiente para presentar correctamente los colores de los objetos contenidos en una región, pero de no ser así, los objetos se pueden dividir horizontalmente en objetos más pequeños, adyacentes verticalmente unos con otros, de forma que cada una de esas regiones tenga una CLUT diferente.

Como ya hemos mencionado, un flujo de subtítulos puede transportar varios servicios de subtítulos. Para estos casos, las páginas de un servicio en concreto se identifican todas con el mismo page_id, formando la **página de composición** de ese servicio (esta página es obligatoria). Además, el sistema de subtitulado permite compartir datos de subtitulado entre los servicios que viajan dentro del mismo flujo de subtítulos. Dichos datos forman parte de la **página auxiliar** (esta página es opcional) y es de especial utilidad cuando existen elementos gráficos comunes en varias páginas. No obstante, normalmente se tiende a transportar los diferentes

servicios en flujos diferentes y PIDs diferentes. En cualquiera de estos casos, el PID, el lenguaje y los page_ids correspondientes al servicio de subtitulado son referenciados por la PMT (ver PMT en 3.2.8.2, y Descriptor de subtitulado en 3.3.10), lo cual puede ser observado en la Figura 3.16.



(a) Mediante page_ids diferentes (Con página auxiliar)

(b) Mediante PIDs diferentes (Sin página auxiliar)

Figura 3.16: Formas de distinguir entre los diferentes idiomas de subtitulado para un mismo servicio [16].

En resumen, el sistema de subtitulado propuesto por DVB proporciona una serie de técnicas que permiten realizar la transmisión eficiente de los datos:

- Los objetos se transmiten sólo una vez, pero pueden ser presentados en una región muchas veces.
- Los objetos que se usan en más de un servicio de subtitulado se transmiten sólo una vez, contenidos en la página auxiliar definida en la norma como *Ancillary Page*.
- Los datos de los píxeles se comprimen mediante el algoritmo de codificación run-length (RLC). En el anexo D se describe gráficamente el funcionamiento de este tipo de compresión.
- Es posible realizar una codificación con menor número de bits por píxel que la indicada por la resolución de un objeto dado. Para esto es necesario incluir una tabla que contenga el mapeo de bits a realizar.

- La definición de los colores de cada entrada de la CLUT se puede realizar de dos formas: usando los 32 bits disponibles o usando únicamente 16 bits. De esta forma tenemos a nuestra disposición la posibilidad de elegir entre un compromiso de precisión de color o de ancho de banda.

En cuanto a los subtítulos destinados a ser presentados en alta definición, es posible incluir una estructura de datos denominada *display_definition_segment* (segmento de definición de presentación), la cual define el tamaño de la pantalla de presentación de los subtítulos. En caso de que el flujo de subtítulos vaya a ser presentado de acuerdo a la definición estándar de TV, los servicios de subtítulos no necesitan incluir un display_definition_segment.

Por último, el estándar ETSI EN 300 743 [16] define otra forma de presentar los subtítulos basada en códigos de caracteres o cadenas de códigos de caracteres, para lo cual es necesario que el decodificador sea capaz de generar el símbolo gráfico de cada uno de los códigos. Las especificaciones relativas a la presentación de subtítulos mediante este método no se definen en la norma y se dejan en manos de los proveedores de contenidos y los fabricantes de decodificadores. Por esta razón no es una práctica usada en España.

3.4.1. Jerarquía de datos

La unidad básica del flujo de subtítulos es el segmento de subtitulado. Estos segmentos son transportados en paquetes PES, que a su vez son transportados en paquetes de transporte.

El juego completo de segmentos de un servicio de subtitulado asociados a un mismo PTS recibe el nombre de *display set*. El último segmento de un display set debe estar seguido de un *end_of_display_set segment*, el cual anuncia que no hay más datos asociados a ese PTS. **Los display sets deben ser entregados en el orden correcto de presentación y deben diferir unos de otros en más de un frame de vídeo (según el PTS)**.

Los datos de subtitulado se dividen en los siguientes segmentos de subtítulos:

- *Display definition segment* - DDS: Este segmento es opcional y sirve para definir explícitamente, el tamaño de la pantalla para la cual ha sido creado el servicio. Siempre que el servicio de subtitulado no esté preparado para ser emitido en el formato estándar de TV, es necesario incluir este segmento.

- *Page composition segment* - PCS: porta información sobre la página de composición del subtítulo: lista de regiones incluidas, posición espacial de cada región, time-out de la página y el estado de la página.

- *Region composition segment* - RCS: contiene información sobre la composición de la región y sus atributos: tamaño de la región, color de fondo, resolución en

píxeles de la región, identificador de la CLUT a usar en la región, así como la lista de objetos incluidos en la región y su posición dentro de la región.

- *CLUT definition segment* - CDS: contiene la información necesaria para describir la CLUT como puede ser el CLUT_id, al que hacen referencia las regiones, y los colores de las entradas de la CLUT.
- *Object data segment* - ODS: este segmento lleva información de un objeto específico. Existen dos tipos de objetos:
  - objetos gráficos: contienen bitmaps codificados mediante RLC.
  - objetos de texto: contienen cadenas de códigos de caracteres.
- *End of display set segment* - EODSS: este segmento no contiene información interna, simplemente se usa para advertir de que no son necesarios más segmentos antes de iniciar la decodificación del display set actual.

Un flujo de subtítulos puede componerse de varios servicios de subtitulado. Sin embargo, un flujo de subtítulos no puede transportar un servicio de subtítulos que contenga un display_definition_segment y otro que no lo tenga. Para estos casos, los servicios de subtitulado deben ir en flujos separados y con PIDs diferentes.

La jerarquía de datos quedaría tal y como se refleja en la Figura 3.17 en la página siguiente, en la cual se empieza a nivel de segmento hasta llegar al flujo de transporte. Para hacer más fácil su comprensión, sólo tenemos en cuenta los paquetes PES de los servicios de subtitulado, aunque en una situación real, éstos viajarían en el flujo de transporte con el resto de paquetes que hemos ido describiendo en este capítulo (paquetes de vídeo, paquetes de audio, paquetes de PSI, etc).

La situación que plantea dicha figura es el caso de dos flujos de subtitulado: uno para TV estándar y otro para HDTV. Puesto que el segundo es en HDTV necesita incluir un display definition segment, sin embargo, el primero puede omitirlo ya que no es necesario. Por otro lado, en el caso del flujo estándar, se ha planteado a modo de ejemplo, la existencia de dos servicios de subtítulos, cada uno en un idioma diferente. Suponiendo que ambos servicios tengan elementos comunes, como puede ser el logotipo de la cadena, una buena opción es incluir una página compartida, es decir, una ancillary page, con el fin de evitar enviar información duplicada. Si además de compartir elementos gráficos, también comparten la CLUT, es decir, tienen la misma tabla de colores, es posible incluir el segmento de definición de la CLUT dentro de la ancillary page[11]. Una vez se han generado los segmentos que forman los servicios de subtitulado, éstos pasan a formar parte de los paquete PES, los cuales terminan formando el flujo de transporte junto con el resto de flujos elementales.

[11] El nombre de página auxiliar no debe llevar a confusión. A pesar de recibir el nombre de página, no contiene ningún PCS ni RCS (hecho que se explica en la siguiente subsección). La forma de ver la ancillary page es como una zona de datos compartida por las páginas de subtítulos.

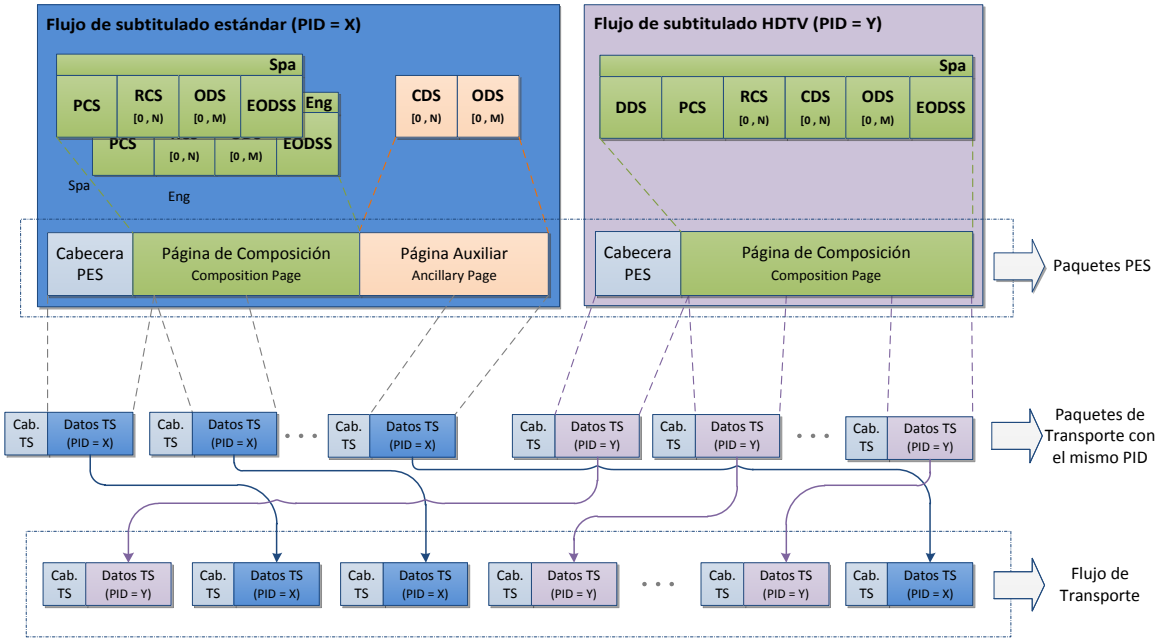


Figura 3.17: Construcción de un flujo de subtítulos (Jerarquía de datos DVB-SUB).

3.4.2. Jerarquía Temporal

A nivel de segmento también se puede definir una jerarquía temporal en la que el nivel más alto está representado por la época. En una época, la página de composición y la región de composición pueden cambiar (por ejemplo, objetos o regiones pueden ser añadidos o eliminados). Haciendo un símil, una época es como una secuencia de vídeo MPEG, por lo que se define la época como la secuencia de una o más páginas. Entre una época y la siguiente no se guarda el estado del decodificador.

Todos los segmentos apuntados por el identificador composition_page_id deben ser entregados antes que los segmentos apuntados por el ancillary_page_id. **El identificador ancillary_page_id no debe apuntar a ningún PCS ni RCS.**

Al inicio de una época, el display set debe incluir el juego completo de RCSs para todas las regiones que van a ser usadas durante una época. El PCS debe contener únicamente el subconjunto de regiones que estarán presentes al inicio de la época. En un caso límite, un PCS puede no presentar regiones visibles[12].

3.4.3. Modelo temporal de decodificación

El uso de memoria por parte del decodificador depende directamente del tamaño y resolución de las regiones contenidas en la página. Al comenzar una época, el

[12]Este es el caso de las páginas de borrado, es decir, aquellas páginas que se utilizan para hacer desaparecer el subtítulo de la pantalla y que se utilizan como transición entre subtítulos consecutivos.

decodificador debe realizar la reserva de memoria que se va a necesitar a lo largo de toda esa época, durante la cual, las páginas pueden pasar por tres estados diferentes: *mode change*, *acquisition point* y *normal case*. El estado de una página queda definido en el PCS, tal y como veremos en la subsección 3.4.7.

3.4.3.1. Mode change

Los límites de las épocas se encuentran delimitados por aquellas páginas cuyo estado es *mode change*. Cuando el decodificador recibe un segmento PCS que indica este estado, todas las reservas de memoria realizadas para los segmentos anteriores son descartadas y el decodificador es reseteado.

Puesto que una página con el estado mode change anuncia el inicio de una época, todas las regiones que se van a usar a lo largo de su existencia deben ser definidas en el primer display set de la época. De esta forma, el decodificador es capaz de planificar el uso de la memoria de antemano. Por esta misma razón, el primer display set debe contener todas las entradas de la CLUT que se van a usar para que el decodificador sepa la cantidad de entradas que van a ser necesarias (su valor puede cambiar durante la época, pero no su cantidad).

3.4.3.2. Acquisition point

El estado de página *acquisition point* indica que se va a realizar una descripción completa del uso de memoria. Sin embargo, el uso de memoria no va a cambiar respecto a la descripción realizada en el estado *mode change*. Cuando el decodificador recibe páginas con este estado, únicamente debe prestar atención al desarrollo de la página y no a la cantidad de memoria que va a ser necesaria.

Mientras que el uso del estado *mode change* puede requerir que el decodificador interrumpa la visualización de los gráficos durante la reasignación de la memoria, el estado *acquisition point* no produce este efecto. De ahí que el estado *mode change* se use con poca frecuencia (por ejemplo al inicio de un programa). Por su parte, el estado *acquisition point* puede ser usado cada pocos segundos facilitando la adquisición rápida de datos por parte de los decodificadores.

3.4.3.3. Normal case

Una página con este estado indica que el juego de RCSs puede no estar completo. En este caso, solo es necesario incluir las regiones cuyos datos hayan cambiado, es decir, las regiones cuyo bitmap o CLUT van a ser modificadas en el display set.

Un display set no tiene por qué contener un PCS. Cuando no se incluye ningún segmento de composición de página, el estado de página no queda definido de manera explícita, sin embargo, implícitamente, el estado es *normal case*.

3.4.3.4. Presentation Time Stamps

Todos los datos de subtitulado están asociados con un PTS para controlar el momento en que los datos decodificados son mostrados, por lo que todos los datos de los subtítulos van contenidos en paquetes PES. Por consiguiente, **por cada servicio de subtitulado solamente puede haber un display set dentro de cada paquete PES**, ya que de lo contrario, significaría que dos display sets de un mismo servicio se tendrían que mostrar al mismo tiempo, lo cual resulta imposible.

En cambio, un paquete PES puede contener los display sets de diferentes servicios de subtitulado, todos ellos compartiendo el mismo tiempo de presentación. Si se da el caso de que los datos de subtitulado superan el límite del tamaño de un paquete PES, entonces los segmentos que comparten un mismo PTS deberán ser transmitidos en más de un paquete PES, todos ellos con el mismo valor de PTS.

En conclusión, **todos los segmentos de un display set deben ser transportados en uno o más paquetes PES que tengan el mismo PTS**.

3.4.4. Modelo de memoria del decodificador

Un PCS con el estado de página *mode change* destruye todas las zonas de memoria borrando los contenidos de la memoria asignada para píxeles y la página de composición. Tal y como ya se ha explicado, estas zonas de memoria persisten hasta que llega una nueva página con estado *mode change*. No hay ningún mecanismo para reasignar parcialmente la memoria durante una época, de ahí la importancia de calcular de forma correcta la cantidad de memoria necesaria.

3.4.4.1. Memoria de píxeles

La memoria asignada en el decodificador para los píxeles tiene un tamaño de 80 kbytes (320 kbytes en el caso de decodificadores capaces de manejar flujos que incluyan DDS). **Esta memoria nunca puede ser desbordada**. Un 75 % de la memoria se asigna para la pantalla actual mientras que el 25 % restante se reserva para pantallas futuras. En la definición que se hace del modelo decodificador de subtítulos, se asume que todas las regiones usadas durante una época son almacenadas en la memoria de píxeles, siendo la reserva de memoria por cada región la siguiente:

$$region\_bits = region\_width \times region\_height \times region\_depth \tag{3.6}$$

Donde *region_depth* es la resolución de píxel expresada en bits para una región en concreto. Así pues, durante una época, **el tamaño de la memoria de píxeles es la suma de la memoria asignada a cada región para esa época**.

3.4.4.2. Memoria de página de composición

Esta memoria contiene toda la información sobre la composición de la página, la composición de las regiones y la definición de la CLUT. Los bytes asignados a esta memoria se describen en la tabla 3.2.

Información	Tamaño necesario
PCS sin la lista de regiones	4 bytes
por región incluida	6 bytes
RCS sin la lista de objetos	12 bytes
por objeto incluido	8 bytes
Definición de la CLUT sin entradas	4 bytes
por entrada con resolución parcial	4 bytes
por entrada con resolución completa	6 bytes

Tabla 3.2: Memoria de página de composición.

3.4.5. Borrado de página

El borrado de página, por definición tiene lugar en el momento indicado por el valor *page_time_out* de la PCS. El borrado de página no afecta al estado de la memoria de píxeles por lo que no existe ningún impacto sobre el renderizado del decodificador de subtítulos.

No obstante, en la mayoría de casos no se utiliza este sistema de borrado ya que presenta un problema: sólo se puede especificar el tiempo de borrado en segundos. Así pues, si se desea una precisión de décimas o centésimas de segundo, hay que recurrir a otra solución. La otra alternativa posible es hacer uso de páginas de borrado, es decir, páginas vacías (sin ninguna región) que hagan desaparecer la página que se está visualizando en pantalla. La ventaja de este sistema es, que al indicar el instante en el que debe aparecer la página de borrado mediante el PTS, disponemos de una resolución mucho mayor, por lo que los resultados que se obtienen ofrecen mayor exactitud.

3.4.6. Modificación de la CLUT

Una vez definida una región, ésta queda ligada de forma permanente a una CLUT. Sin embargo, esa CLUT puede redefinirse, es decir, los colores asociados a cada código de píxel puede ser modificados. Cuando la CLUT cambia, no implica ninguna carga adicional al proceso de renderizado del subtítulo.

3.4.7. Formato del paquete PES para subtitulado

Al igual que los datos de vídeo y de audio, los subtítulos DVB viajan en paquetes PES cumpliendo la misma sintaxis y semántica que la definida en la norma ISO/IEC 13818-1 [14] pero con las siguientes restricciones:

- *stream_id*: su valor es "0b1011 1101" ("0xBD"), lo que indica *private_stream_1*.
- *data_alignment_indicator*: flag que se fija a 1 indicando que los segmentos de subtítulos están alineados con los paquetes PES.
- *PES_packet_data_byte*: el campo de datos presenta las características que se describen en el apartado 3.4.7.1 y siguientes.

3.4.7.1. Sintaxis y semántica del campo de datos de un paquete PES de subtitulado

El paquete PES para subtitulado transporta el flujo de subtítulos DVB codificados dentro del campo de datos según se define en la Figura 3.18:

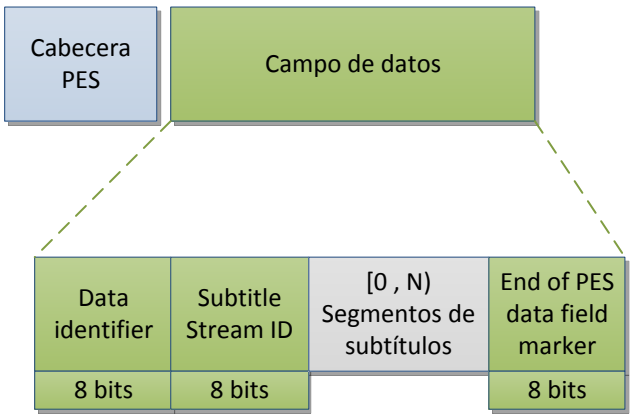


Figura 3.18: Campo de datos de un paquetes PES de Subtitulado.

- *data_identifier*: su valor para el flujo de subtítulos DVB es "0x20".
- *subtitle_stream_id*: el flujo de subtítulos DVB es identificado con el valor "0x00".
- *end_of_PES_data_field_marker*: campo de 8 bits cuyo contenido es siempre "0xFF".

3.4.7.2. Sintaxis y semántica de un segmento de subtitulado

El segmento es la unidad básica de los flujos de subtítulos. Todos los segmentos presentan una parte común formada por cuatro campos cuya estructura se describe en la Figura 3.19.

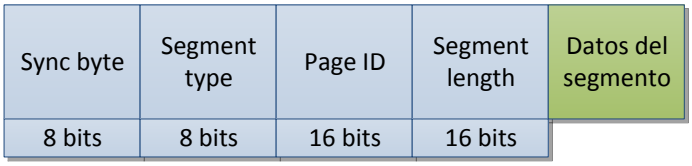


Figura 3.19: Segmento de Subtitulado.

- *sync_byte*: campo de 8 bits codificado como "0x0F". Sirve para verificar que la longitud del segmento se corresponde con la indicada por el campo *segment_length*. Cuando el decodificador encuentra este valor tras terminar de leer un segmento, interpreta que a continuación viene otro segmento.
- *segment_type*: este campo indica el tipo de datos que transporta el segmento. La Tabla 3.3 contiene todos sus valores posibles.

segment_type	Tipo de segmento
0x10	Page composition segment
0x11	Region composition segment
0x12	CLUT definition segment
0x13	Object data segment
0x14	Display definition segment
0x40 - 0x7F	Reservado para uso futuro
0x80	End of display segment
0x81 - 0xEF	Datos privados
0xFF	Relleno
Resto de valores	Reservado para uso futuro

Tabla 3.3: Tipos de Segmentos.

- *page_id*: este campo se relaciona directamente con la información de los descriptores de subtítulos. Identifica al servicio de subtitulado al que pertenecen

los datos de este segmento. Si el valor del page_id es el mismo que el definido en el descriptor de subtitulado como composition_page_id, este segmento transporta datos específicos de un servicio de subtitulado. En cambio, si es el mismo que el de la ancillary_page_id, contiene datos compartidos por múltiples servicios de subtitulado.

- *segment_length*: especifica el número de bytes del segmento a partir de este campo.

Una vez que se ha visto la parte común de todos los segmentos, procedemos a describir las características que diferencian a unos de otros.

3.4.7.3. Display definition segment

Es el segmento encargado de realizar la definición de las dimensiones de la pantalla de subtítulos. **En caso de que no se incluya este segmento, se asume que las dimensiones son las de la televisión estándar, es decir, 720 píxeles de ancho por 576 líneas de alto**. Este segmento presenta la estructura descrita por la Figura 3.20, cuyos campos se describen a continuación:

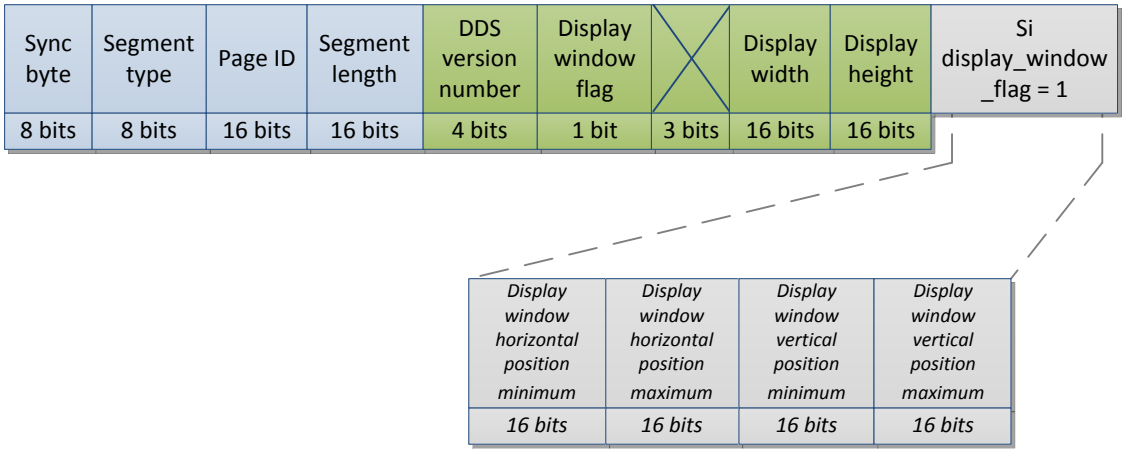


Figura 3.20: Display definition segment.

- *dds_version_number*: indica la versión del segmento. Cuando alguno de los elementos del *display definition segment* cambia, el número de versión es incrementado. Al igual que la gran mayoría de contadores usados en los segmentos que se van a describir de aquí en adelante, se trata de un **contador módulo 16**, es decir, cuenta de 0 a 15, y al llegar a 15, vuelve a empezar en 0, y así sucesivamente.

- *display_window_flag*: es un indicador de 1 bit. Cuando su valor es "1", el display set de subtítulos asociado a este DDS es renderizado dentro de una ventana contenida dentro de la pantalla descrita por el DDS. Las dimensiones de la ventana quedarían definidas por los últimos cuatro campos de este segmento. En cambio, si display_window_flag = 0, los subtítulos asociados al DDS son renderizados directamente dentro de las dimensiones de la pantalla definida mediante los campos *display_width* y *display_height*.

- *display_width*: especifica el ancho máximo de la pantalla en píxeles menos 1. El valor indicado por este campo debe estar entre 0 y 4095.

- *display_height*: especifica el alto máximo de la pantalla en líneas menos 1. El valor indicado por este campo debe estar entre 0 y 4095.1

- *display_window_horizontal_position_minimum*: especifica la posición del píxel situado más a la izquierda respecto al borde izquierdo de la pantalla.

- *display_window_horizontal_position_maximum*: especifica la posición del píxel situado más a la derecha respecto al borde izquierdo de la pantalla.

- *display_window_vertical_position_minimum*: especifica la posición del píxel situado en la posición más alta respecto al borde superior de la pantalla.

- *display_window_vertical_position_maximum*: especifica la posición del píxel situado en la posición más baja respecto al borde superior de la pantalla.

3.4.7.4. Page composition segment

La composición de la página asociada a un servicio de subtítulos es transportada en este tipo de segmentos. El valor del *page_id* debe ser el del *composition_page_id* especificado en el descriptor del subtítulo, ya que tal y como se mencionó en la subsección 3.4.2, el *ancillary_page_id* no puede apuntar a ningún PCS ni RCS. La Figura 3.21 muestra la estructura del PCS en la cual se puede apreciar como última parte, la lista de regiones que componen la página. A priori, el decodificador no conoce de cuantas regiones se compone la página, por lo que la forma en que procesa estos datos es teniendo en cuenta el tamaño del segmento. De esta forma, sabe en todo momento cuándo ha llegado al final del PCS.

- *page_time_out*: es el tiempo, expresado en segundos, tras el cual la página es borrada de la pantalla, a no ser que se redefina posteriormente o tenga lugar una nueva instancia de página (este es el caso de las páginas de borrado).

- *page_version_number*: contador módulo 16 que indica la versión de la página. Cuando alguno de los contenidos de la página cambia, su valor es incrementado.

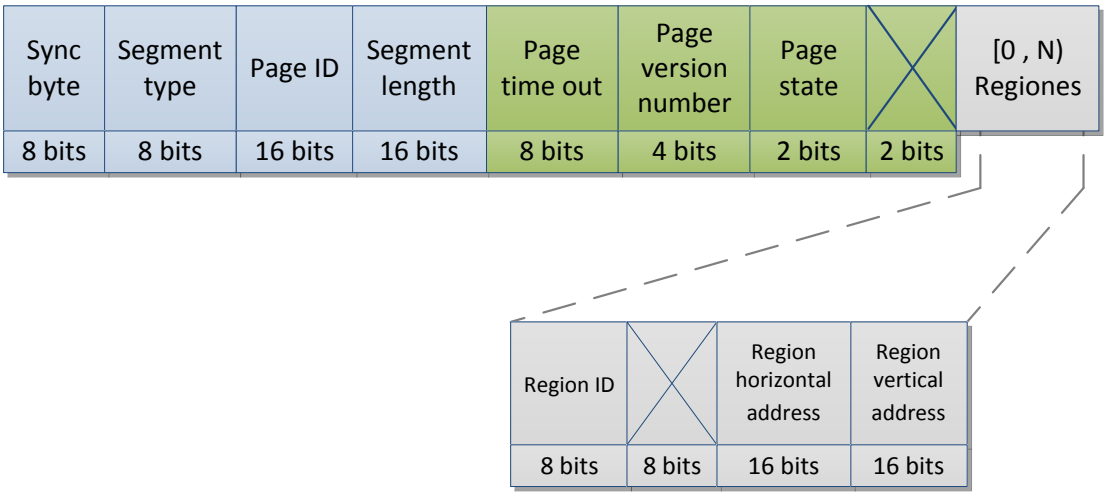


Figura 3.21: Page composition segment.

- *page_state*: indica el estado de la página. Sus posibles valores se definen en la Tabla 3.4:

Valor	Estado de la página	Efecto en la página	Descripción
00	*normal case*	Actualización de la página	El display set contiene solo los segmentos que han cambiado
01	*acquisition point*	Refresco de la página	El display set contiene todos los segmentos
10	*mode change*	Nueva página	El display set contiene todos los segmentos
11	reservado		Reservado para uso futuro

Tabla 3.4: Valores posibles de page_state [16].

- *region_id*: identifica a una región dentro de la página. Sólo las regiones que aparecen aquí listadas son mostradas en la página. Las regiones deben aparecer en orden ascendente del campo *region_vertical_address*.

- *region_horizontal_address*: especifica la dirección horizontal del píxel izquierdo superior de la región. Los píxeles se direccionan de izquierda a derecha empezando por el valor 0.

- *region_vertical_address*: especifica la dirección vertical de la línea superior de

la región. Las líneas se direccionan de arriba hacia abajo empezando por el valor 0.

3.4.7.5. Region composition segment

Este segmento contiene la lista de objetos que componen la región, posicionados de tal forma que no se superpongan. Los campos que lo forman son los siguientes:

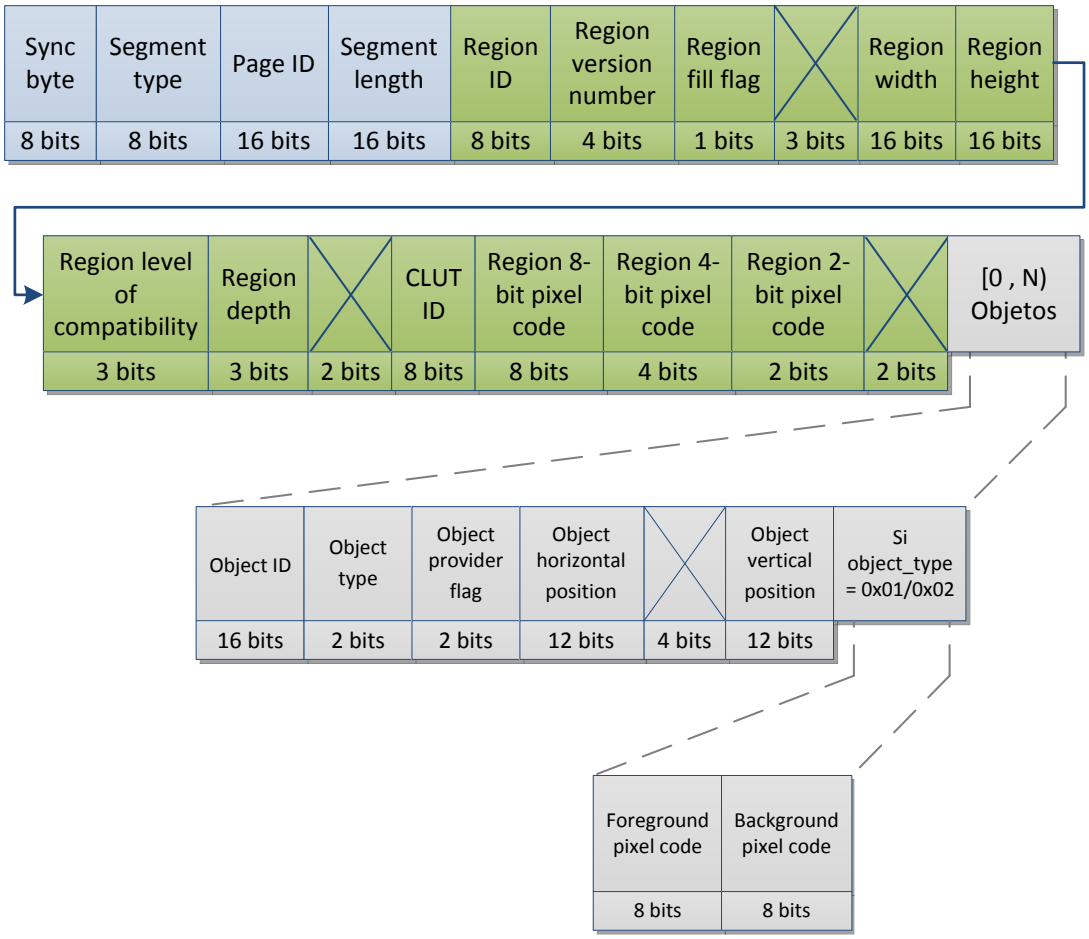


Figura 3.22: Region composition segment.

- *region_id*: identifica a la región cuya descripción se realiza en este segmento.
- *region_version_number*: contador módulo 16 que indica la versión de la región. Se incrementa en las siguientes situaciones:
  - *region_fill_flag* = 1.
  - La CLUT ha sido modificada.

- La región contiene una lista no vacía de objetos.

- *region_fill_flag*: cuando vale "1", indica que el fondo de la región es rellenado con el color definido por el campo *region_n-bit_píxel_code*.

- *region_width*: especifica el ancho de la región en píxeles. Para servicios de subtitulado en los que no se incluye un DDS, este valor debe estar entre 1 y 720 ($(display\_width + 1)$, si se incluye una DDS), y la suma del ancho de la región y de la *region_horizontal_address* no debe ser mayor de 720 ($(display\_width + 1)$, si se incluye una DDS). El valor típico de este campo cuando se trata de subtítulos es 720 ("0x02D0").

- *region_height*: especifica el alto de la región en píxeles. Para servicios de subtitulado en los que no se incluye un DDS, este valor debe estar entre 1 y 576 ($(display\_height + 1)$, si se incluye una DDS), y la suma del ancho de la región y de la *region_vertical_address* no debe ser mayor de 576 ($(display\_height + 1)$, si se incluye una DDS). El valor típico de este campo cuando se trata de subtítulos es 44 ("0x002C").

- *region_level_of_compatibility*: indica el tipo mínimo de CLUT necesario en el decodificador para decodificar esta región. Si el decodificador no soporta este requisito, la región no es mostrada. La Tabla 3.5 contiene los tipos permitidos y el valor del campo asociado a esos tipos.

Valor	Descripción
0x00	Reservado
0x01	Entradas de 2-bit en la CLUT requeridas
0x02	Entradas de 4-bit en la CLUT requeridas
0x03	Entradas de 8-bit en la CLUT requeridas
0x04 - 0x07	Reservado

Tabla 3.5: Valores posibles de *region_level_of_compatibility* [16].

- *region_depth*: identifica la resolución de píxeles de esta región según los valores de la Tabla 3.6.

Valor	Descripción
0x00	Reservado
0x01	2-bit
0x02	4-bit
0x03	8-bit
0x04 - 0x07	Reservado

Tabla 3.6: Valores posibles de *region_depth* [16].

- *CLUT_id*: identificador que hacer referencia a la CLUT que se va a usar en esta región.

- *region_8-bit_pixel-code*: especifica la entrada de la CLUT de 8 bits que se va a usar como color de fondo de la región, siempre que esté puesto el *region_fill_flag* y el valor de *region_depth* sea "0x03" (8 bits). Su valor es indefinido cuando la resolución es de 2 o 4 bits.

- *region_4-bit_pixel-code*: especifica la entrada de la CLUT de 4 bits que se va a usar como color de fondo de la región cuando el *region_fill_flag* vale "1", si el *region_depth* es de 4 bits, o si es de 8 bits, siempre que la *region_level_of_compatibility* especifique una entrada de CLUT de 4 bits entre los requisitos mínimos. En cualquier otro caso el valor de este campo es indefinido.

- *region_2-bit_pixel-code*: especifica la entrada de la CLUT de 2 bits que se va a usar como color de fondo de la región cuando el *region_fill_flag* vale "1", si el *region_depth* es de 2 bits, o si se da el caso de que vale 8 o 4 bits, siempre que la *region_level_of_compatibility* especifique una entrada de CLUT de 2 bits entre los requisitos mínimos. En cualquier otro caso el valor de este campo es indefinido.

- *object_id*: identificador del objeto que se pretende mostrar en la región.

- *object_type*: especifica el tipo de objeto de acuerdo a los valores expuestos en la Tabla 3.7.

Valor	Tipo de objeto
0x00	Objeto básico. Bitmap
0x01	Objeto básico. Carácter
0x02	Objeto compuesto. Cadena de caracteres
0x03	Reservado

Tabla 3.7: Valores posibles de *object_type* [16].

- *object_provider_flag*: flag de 2 bits que indica cómo se proporciona el objeto. La Tabla 3.8 muestra sus posibles valores.

Valor	Descripción
0x00	Proporcionado en el flujo de subtítulos
0x01	Proporcionado por una ROM en el decodificador
0x02	Reservado
0x03	Reservado

Tabla 3.8: Valores posibles de *object_provider_flag* [16].

- *object_horizontal_position*: especifica la posición horizontal del píxel superior izquierdo del objeto, expresada en número de píxeles desde el borde izquierdo de la región. Este valor debe estar contenido dentro de la región.
- *object_vertical_position*: especifica la posición vertical del píxel superior izquierdo del objeto, expresada en número de líneas desde el borde superior de la región. Este valor debe estar contenido dentro de la región.
- *foreground_píxel_code*: especifica la entrada de la CLUT de 8 bits que ha sido seleccionada como color de los caracteres. Este campo y el siguiente sólo aparecen cuando el valor del campo *object_type* es "0x01" o "0x02".
- *background_píxel_code*: especifica la entrada de la CLUT de 8 bits que ha sido seleccionada como color de fondo de los caracteres.

3.4.7.6. CLUT Definition segment

Es el segmento encargado de definir los colores que se van a aplicar en una familia de CLUTs. A continuación se explica cada uno de sus campos:

- *CLUT-id*: identificador de la CLUT cuyos datos están contenidos en este segmento.

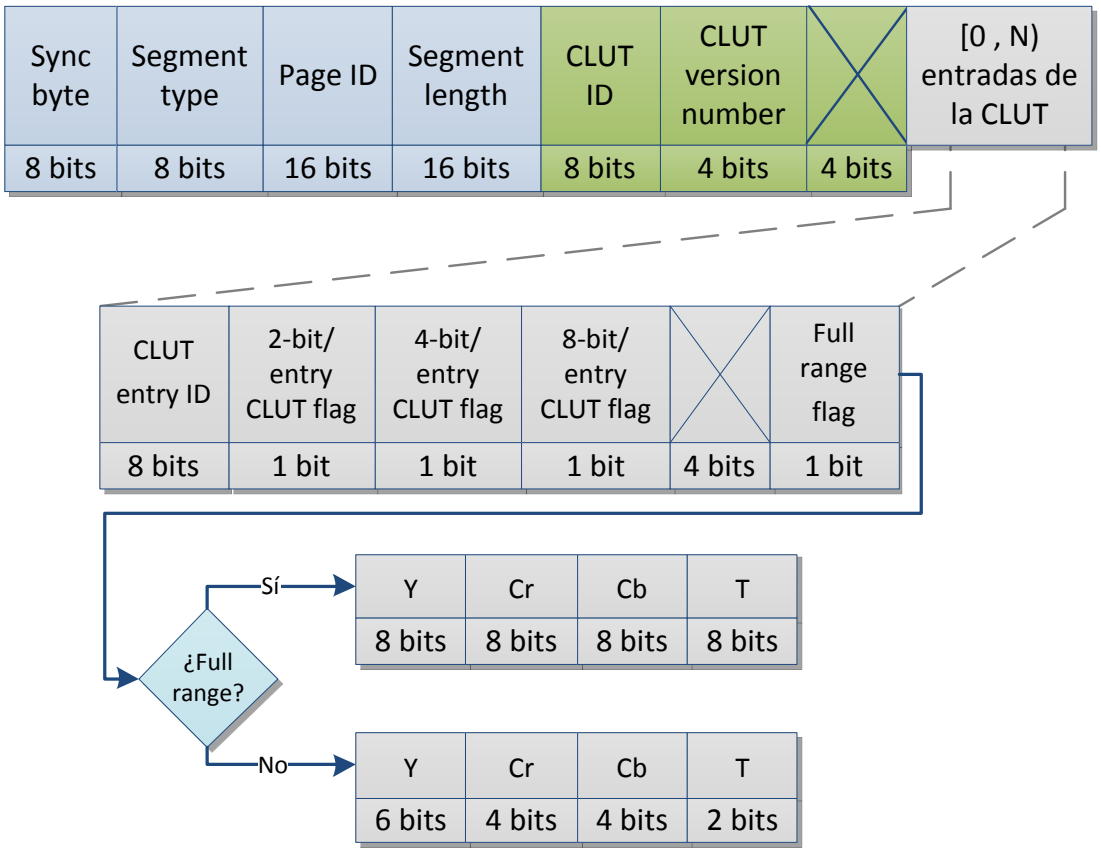


Figura 3.23: CLUT Definition segment.

- *CLUT_version_number*: contador módulo 16 que indica la versión de la CLUT. Cuando alguno de los contenidos de este segmento cambia, el contador es incrementado.

- *CLUT_entry_id*: indica el número de entrada de la CLUT. La primera entrada de la CLUT empieza en 0.

- *2-bit/entry_CLUT_flag*: si vale "1", indica que esta entrada de la CLUT sera cargada en la CLUT de 2 bits.

- *4-bit/entry_CLUT_flag*: si vale "1", indica que esta entrada de la CLUT sera cargada en la CLUT de 4 bits.

- *8-bit/entry_CLUT_flag*: si vale "1", indica que esta entrada de la CLUT sera cargada en la CLUT de 8 bits[13].

[13]Solo uno de los campos N-bit/entry_CLUT_flag debe valer "1" por cada entrada.

- *full_range_flag*: si vale 1, indica que los campos *Y_value*, *Cr_value*, *Cb_value* y *T_value* tienen toda la resolución posible, es decir, 8 bits. En caso contrario, contienen únicamente los bits más significativos.

- *Y_value*: representa la iluminancia de la entrada de la CLUT correspondiente. Si vale cero indica que el color es completamente transparente, y en tal caso, los campos *Cr_value*, *Cb_value* y *T_value* son irrelevantes por lo que deben ser "0".

- *Cr_value*: es la crominancia roja de la entrada en cuestión.

- *Cb_value*: es la crominancia azul de la entrada en cuestión.

- *T_value*: representa el valor de transparencia de la entrada de la CLUT. Si vale "0" indica que no hay transparencia mientras que su valor máximo más uno correspondería a transparencia completa. Para el resto de valores, el nivel de transparencia se obtiene mediante una interpolación lineal. La máxima transparencia se consigue poniendo a cero el campo *Y_value.*

3.4.7.7. Object data segment

Es el segmento que contiene los datos del objeto. Para los objetos gráficos, es decir, aquellos codificados como píxeles, hay que tener en cuenta los siguientes aspectos:

- Se asume que los objetos están entrelazados, es decir, tienen un campo superior y otro inferior, referidos en la norma como *top_field* y *bottom_field*.

- El primer píxel de la primera línea del campo superior es el píxel superior izquierdo del objeto.

- El primer píxel de la primera línea del campo inferior es el píxel situado en la parte izquierda de la segunda línea del objeto.

- Ambos campos, el superior y el inferior, deben presentar bloques denominados como *pixel-data_sub-blocks* y que explicaremos en las líneas siguientes.

- En caso de que este segmento no lleve datos asociados al campo inferior, entonces el pixel-data_sub-block del campo superior es utilizado para el campo inferior.

La estructura de este segmento es algo más compleja que la del resto de segmentos, especialmente, como consecuencia del campo pixel-data_sub-block. Por ello, primeramente vamos a analizar la sintaxis global del segmento para a continuación centrarnos en este campo en detalle.

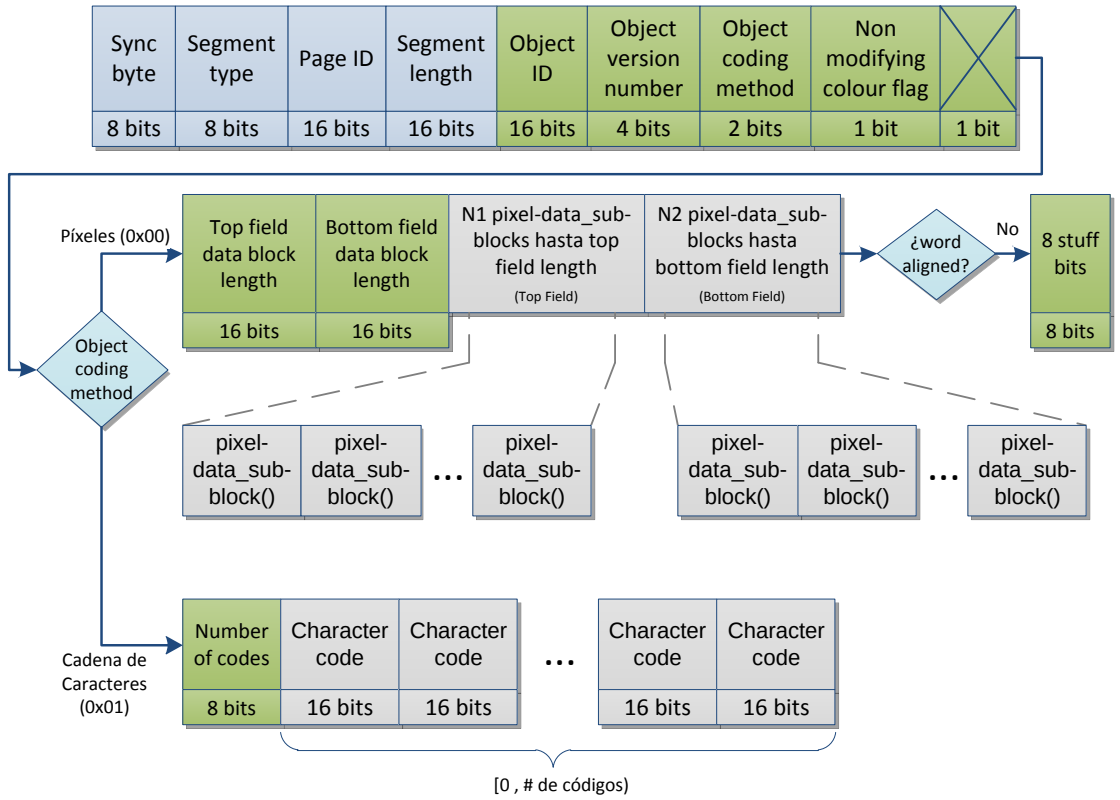


Figura 3.24: Object Data segment.

- *object_id*: identificador de 16 bits del objeto dentro de la página, cuyos datos se describen en este segmento.
- *object_version_number*: contador módulo 16 que indica la versión de los datos de este segmento. Cuando alguno de los contenidos de este segmento cambia, el contador es incrementado.
- *object_coding_method*: especifica el método usado para codificar el objeto. Actualmente solo existen dos posibilidades:
  - 0x00: codificación basada en píxeles.
  - 0x01: codificación basada en cadenas de caracteres.
- *non_modifying_colour_flag*: cuando este indicador vale "1" señala que la entrada 1 de la CLUT representa un color que no se va a modificar. Cuando se asigna este color a un píxel del objeto, entonces los píxeles del fondo de la región subyacente no serán modificados.

- *top_field_data_block_length*: especifica el número de bytes contenidos en total en los *pixel-data_sub-blocks* del campo superior.

- *bottom_field_data_block_length*: especifica el número de bytes contenidos en total en los *pixel-data_sub-blocks* del campo inferior. Si su valor es "0", entonces los datos del campo superior serán los que se usarán para el campo inferior. Esto implica que los objetos de una única línea no estarán permitidos[14].

- *8_stuff_bits*: byte de relleno codificado como "0b00000000" que es usado para conseguir alineación de palabra al final del segmento.

- *number_of_codes*: especifica el número de códigos de caracteres en la cadena cuando el campo *object_coding_method* vale "0x01".

- *character_code*: especifica el código del carácter a través del número de índice en la tabla de caracteres identificada en el descriptor del subtítulo.

Para la codificación basada en píxeles, resulta de vital importancia comprender cómo funcionan los bloques definidos como *pixel-data_sub-blocks*. Estos bloques son las unidades en las que se subdividen, tanto el campo superior, como el inferior, y cuya estructura se describe en la Figura 3.25 en la página siguiente.

El significado de los campos que forman estos bloques es el siguiente:

- *data_type*: identifica el tipo de información contenida en el pixel-data_sub-block de acuerdo a la Tabla 3.9.

Valor	Descripción
0x10	2-bit/píxel code string
0x11	4-bit/píxel code string
0x12	8-bit/píxel code string
0x20	2_to_4-bit_map-table data
0x21	2_to_8-bit_map-table data
0x22	4_to_8-bit_map-table data
0xF0	Código de fin de línea de objeto
	Reservado

Tabla 3.9: Valores posibles del campo *data_type* [16].

[14]En realidad están permitidos, pero van a sufrir efectos flicker al ser mostrados en pantallas de visionado entrelazado.

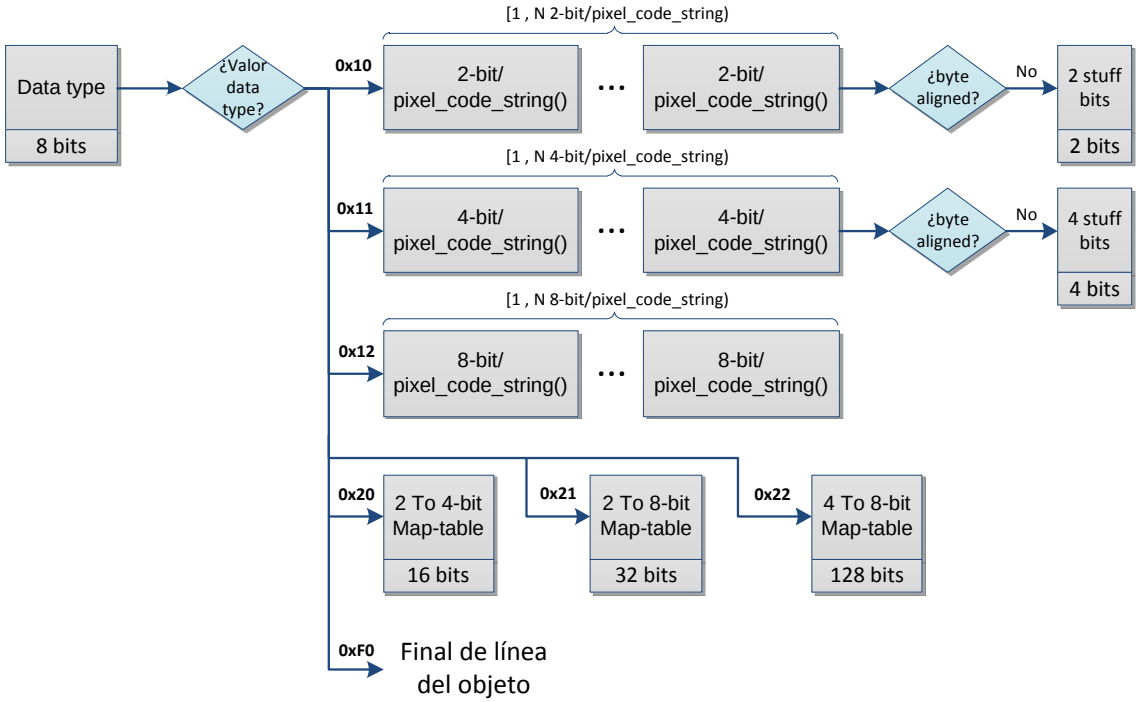


Figura 3.25: Pixel-data sub-block.

- *2_to_4-bit_map-table*: especifica cómo realizar la conversión (mapeo) de los códigos de 2 bits a los códigos de 4 bits de una CLUT, enumerando las 4 entradas de 4 bits a las que van a hacer referencia, comenzando por la entrada 0 y terminando en la entrada 3.

- *2_to_8-bit_map-table*: especifica cómo realizar la conversión de los códigos de 2 bits a los códigos de 8 bits de una CLUT, enumerando las 4 entradas de 8 bits a las que van a hacer referencia, comenzando por la entrada 0 y terminando en la entrada 3.

- *4_to_8-bit_map-table*: especifica cómo realizar la conversión de los códigos de 4 bits a los códigos de 8 bits de una CLUT, enumerando las 16 entradas de 8 bits a las que van a hacer referencia, comenzando por la entrada 0 y terminando en la entrada 15.

- *2_stuff_bits*: campo de dos bits codificado como “0b00” que será insertado cuando sea necesario realizar alineación de bytes tras una secuencia de 2-bit/píxel_code_strings().

- *4_stuff_bits*: campo de cuatro bits codificado como “0b0000” que será inser-

tado cuando sea necesario realizar alineación de bytes tras una secuencia de 4-bit/píxel_code_strings().

Como se puede apreciar en la Figura 3.25 en la página anterior, un pixel-data sub-block está constituido a su vez por unidades de *N-bit/píxel_code_strings()*, de las cuales existen tres tipos, dependiendo del valor del data_type[15].

Antes de continuar adentrándonos en la compleja semántica del segmento de datos de objeto, y de forma más concreta en los *N-bit/píxel_code_strings()*, es preciso hacer un inciso para que se entienda de forma clara el modo en que una imagen puede ser codificada mediante el algoritmo RLC descrito en la norma. Este es el propósito que persigue la Figura 3.26, en la cual se puede observar de forma simplificada y esquematizada, cómo se realizaría esta codificación[16].

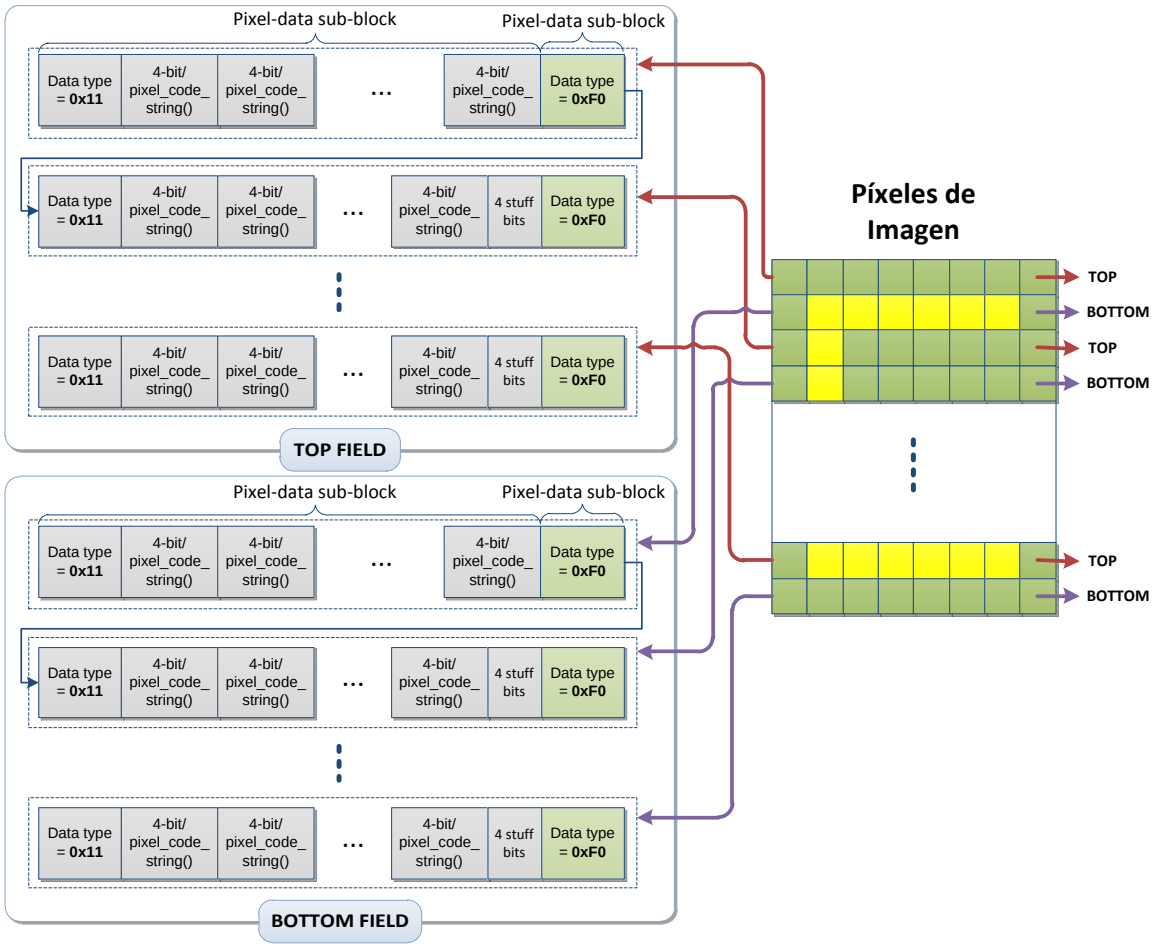


Figura 3.26: Ejemplo de codificación basada en píxeles.

[15]Estos datos dependen directamente del tipo de CLUT asociado a la región

[16]La figura describe un ejemplo simplificado de la codificación utilizada como base para el proyecto. No obstante, existen más posibilidades tal y como describe la Tabla 3.9 en la página 71.

Durante la fase del estudio del arte, tras analizar la forma en que las cadenas de televisión realizan dicha codificación, se concluyó que una CLUT de entradas de 4 bits era el tipo más adecuado para cumplir con los objetivos del proyecto. Así pues, de aquí en adelante nos vamos a centrar en los 4-bit/píxel_code_strings(), ya que es el tipo de cadenas que se va a usar a lo largo del proyecto.

Volviendo a la Figura 3.26 en la página anterior, en ella podemos observar a la izquierda, una imagen dividida en píxeles. De acuerdo a lo que hemos definido previamente, el primer píxel de la primera línea del campo superior es el píxel superior izquierdo del objeto, mientras que el primer píxel de la primera línea del campo inferior es el píxel situado en la parte izquierda de la segunda línea del objeto. En otras palabras, las líneas impares que constituyen la imagen, pertenecen al campo superior (Top field) y las líneas pares, al campo inferior (Bottom field). Todas las líneas del top field son codificadas de forma consecutiva mediante pixel-data_sub-blocks y a continuación, se repite el mismo proceso con las líneas pertenecientes al bottom field.

Si ahora prestamos atención a la forma en que quedan representadas las líneas, éstas presentan siempre la misma estructura: un pixel-data_sub-block() formado por 4-bit/píxel_code_strings(), que de forma conjunta representan una línea del objeto, y un pixel-data_sub-block(), indicando simplemente el final de línea del objeto, es decir, un pixel-data_sub-block() constituido por el campo data_type cuyo valor es "0xF0". De forma opcional, dependiendo de si después de la secuencia de 4-bit/píxel_code_strings(), los bits están alineados o no, habría que incluir los 4 bits de relleno que se pueden observar en la figura a modo de ejemplo.

Por último, vamos a explicar la sintaxis y el significado de los campos que componen los 4-bit/píxel_code_strings(). En esencia, en ellos es donde va contenida la información necesaria para identificar el color de los píxeles. La Figura 3.27 en la página siguiente, describe los campos que forman un 4-bit/píxel_code_string() así como las secuencias de bits posibles.

En algunas de dichas secuencias aparecen conjuntos de C's, que indican el número de entrada de la CLUT que contiene el color correspondiente al grupo de píxeles, y de R's, que constituyen un número que, sumado al número situado encima de ellos, indican la cantidad de píxeles a codificar con ese color. Si interpretamos la figura como si de un árbol se tratase, en función del contenido de los primeros 4 bits y de los campos *switch_1*, *switch_2* y *switch_3*, recorreremos las ramas que constituyen la imagen abarcando todas esas secuencias de bits. El significado del resto de campos que aparecen en la figura se describe a continuación:

- *4-bit_pixel-code*: código de 4 bits especificando el pseudo-color de un píxel como el número de la entrada de la CLUT de 16 entradas o el número de la entrada de una tabla de mapeo de entradas (como las que se describen en la Tabla 3.9 en la página 71).

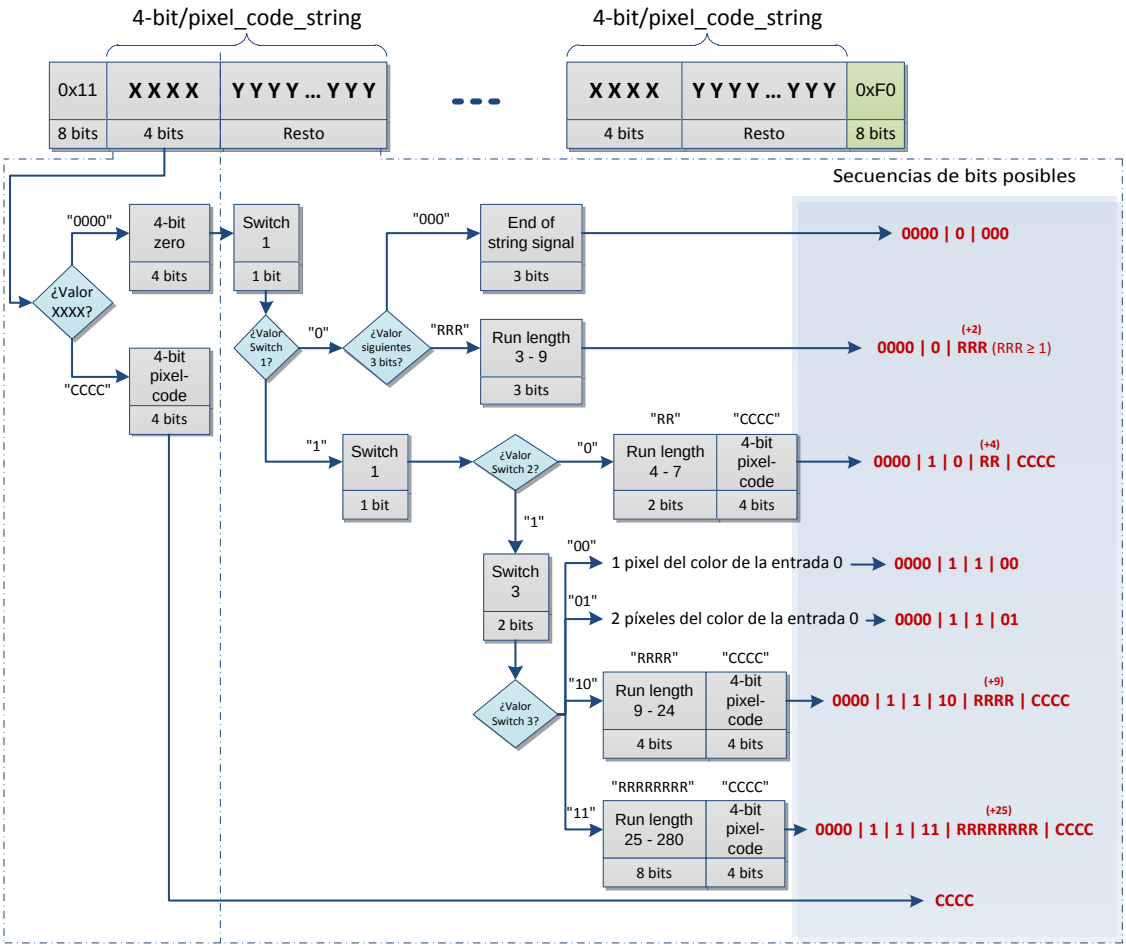


Figura 3.27: 4-bit/píxel_code_string().

- *4-bit_zero*: campo de 4 bits cuyo valor es "0b0000". Carece de significado, simplemente sirve para "continuar por la otra parte del árbol".

- *run_length_3-9*: campo de 3 bits cuyo valor indica el número de píxeles menos 2 que deben ser puestos al color representado por la entrada 0 de la CLUT. Dicho de otro modo, el número indicado por este campo más 2, es el número de píxeles consecutivos de la imagen cuyo color es el pseudo-color de la entrada 0.

- *end_of_string_signal*: campo de 3 bits fijados al valor "0b000". Cuando este campo aparece, advierte el final de la secuencia de 4-bit/píxel_code_strings que describen la línea actual del objeto.

- *run_length_4-7*: campo de 2 bits. Su valor más 4, es el número de píxeles que deben ser puestos del pseudo-color indicado por el pixel-code que le sigue.

- *run_length_9-24*: campo de 4 bits. Su valor más 9, es el número de píxeles

que deben ser puestos del pseudo-color indicado por el pixel-code que le sigue.

- *run_length_25-280*: campo de 8 bits. Su valor más 25, es el número de píxeles que deben ser puestos del pseudo-color indicado por el pixel-code que le sigue.

Así pues, es aquí donde se realiza la codificación RLC propiamente dicha. Tal y como se ha descrito, es muy parecido al algoritmo explicado en el anexo D en la página 239, pero con las características que se acaban de describir. Constituye una buena forma de ahorrar bits, es decir, ancho de banda, en especial, cuando se trata de píxeles que hacen referencia a la entrada 0, la cual habitualmente es ocupada por el color que representa la transparencia. Para terminar con la descripción de este segmento, en el anexo E en la página 241 se incluye una tabla resumen, la cual puede constituir una buena guía a la hora de interpretar la información contenida en los bits de un pixel-data_sub-block().

3.4.7.8. End of display set segment

Este segmento indica al decodificador que la transmisión del display set se ha completado. Debe ser insertado en el flujo inmediatamente después del último ODS del display set. Su estructura es la más sencilla de todas pues solo se compone de los campos comunes que hay al principio de todos los segmentos. Respecto al valor del campo *page_id*, éste será el mismo que el de *ancillary_page_id,* siempre y cuando el servicio de subtitulado use datos compartidos. En cualquier otro caso, su valor será el del campo *composition_page_id*.

3.5. Digital Video Broadcasting (DVB); Specification for conveying ITU-R System B Teletext in DVB bitstreams. ETSI EN 300 472

Esta norma especifica el método por el cual los subtítulos en formato Teletexto, conocidos como *EBU Teletext*, pueden ser transportados en los flujos DVB. Este mecanismo de transporte debe satisfacer los siguientes requisitos:

- Soportar la transcodificación de los datos de Teletexto dentro del *Vertical Blanking Interval* (VBI) del vídeo analógico. La señal transcodificada debe ser compatible con los decodificadores de Teletexto de los receptores de TV existentes.

- La máxima tasa de datos para cada servicio de Teletexto debe ser equivalente a 16 líneas por campo, de tal forma que el servicio siempre esté disponible para la transcodificación dentro del VBI.

- El mecanismo de transmisión debe ser capaz de transmitir subtítulos con precisión temporal respecto al vídeo.

La mayoría de los canales de TV en España transmiten subtítulos en este formato: La 1, La 2, Telecinco, Cuatro, La Sexta, etc. Este es el formato de compatibilidad con la televisión analógica, razón por la cual en algunas cadenas conviven estos subtítulos con los flujos DVB-SUB.

3.5.1. Inserción del Teletexto dentro del múltiplex de transporte MPEG-2

Al igual que el resto de flujos, los datos de Teletexto, entre los cuales se encuentran los subtítulos en formato Teletexto, son enviados en paquetes PES transportados en flujos de transporte. El PID de los paquetes de transporte asociados a un servicio debe estar indicado en la PMT de dicho servicio, en la que se especificará que el tipo de flujo transportado son datos privados, para lo cual el stream_type debe valer "0x06". Al igual que vimos en la subsección 3.3.10 en la página 49, la entrada de la PMT correspondiente al PID de los paquetes de Teletexto, debe contener el descriptor adecuado del flujo, que en este caso sera el descriptor de Teletexto. A continuación se explican sus características.

3.5.1.1. Teletext Descriptor

Es un descriptor muy parecido al de subtitulado pero que presenta algunas salvedades. En la Figura 3.28 se puede observar la estructura del descriptor, cuyos campos son los siguientes [15]:

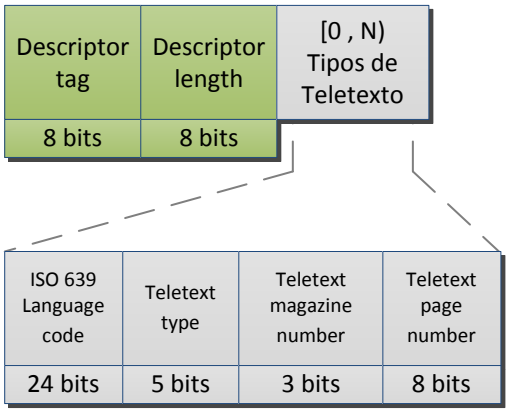


Figura 3.28: Descriptor de Teletexto.

- descriptor_tag: Para teletexto este campo vale "0x56".

- descriptor_length: indica el tamaño del descriptor a partir de este campo.
- ISO_639_language_code: es el código de 24 bits que contiene los 3 caracteres del código del idioma.
- *teletext_type*: campo de 5 bits que indica el tipo de página de Teletexto. Su valor puede ser cualquiera de los contenidos de la Tabla 3.10.

Valor	Descripción
0x00	Reservado para uso futuro.
0x01	Página inicial de Teletexto.
0x02	Página de subtítulos de Teletexto.
0x03	Página de información adicional.
0x04	Página de planificación de programas.
0x05	Página de subtítulos de Teletexto para personas sordas.
0x06 to 0x1F	Reservado para uso futuro.

Tabla 3.10: Valores posibles del campo teletext_type [16].

- *teletext_magazine_number*: es un campo de 3 bits que identifica el número de programa tal y como se define en la norma EN 300 706.
- *teletext_page_number*: campo de 8 bits que proporciona dos dígitos hexadecimales de 4 bits que identifican el número de página tal y como se define en la norma EN 300 706.

Un servicio puede incluir más de un flujo de datos de Teletexto, dado que cada flujo presenta un valor diferente del campo *data_identifier*, y que los flujos son diferenciados teniendo en cuenta sus respectivos descriptores de Teletexto contenidos en los datos PSI.

3.5.2. Formato del paquete de transporte para datos de Teletexto

Los paquetes de transporte para este tipo de flujos son iguales que los que se han descrito previamente, pero con la siguiente restricción: el campo adaptation_field_control solo puede valer “0b01” o “0b10”, es decir, los paquetes solo van a tener datos o campo de adaptación, nunca van a ir mezclados.

3.5.3. Formato del paquete PES para datos de Teletexto

Los paquetes PES presentan las siguientes características particulares:

- stream_id: al igual que el flujo de subtítulos DVB-SUB, vale "0xBD", es decir, indica que es un *private_stream_1*.

- *PES_packet_length*: su valor debe estar siempre puesto a $(N \times 184) - 6$, donde N es un número entero, de tal forma que el paquete PES termine siempre al final de un paquete de transporte.

- Data_alignment_indicator: estará puesto a 1 indicando que las unidades de acceso de Teletexto están alineadas respecto a los paquetes de transporte.

- *PES_header_data_length*: su valor se fija a "0x24", independientemente de si se incluyen el PTS y otros campos opcionales en la cabecera.

- stuffing_byte: la cabecera del paquete PES presenta tantos bytes de relleno como sean necesarios para llegar hasta el valor indicado por el campo PES header data length, de forma que toda la cabecera del paquete PES tenga siempre un tamaño de 45 bytes.

- *PES_packet_data_byte*: estos bytes se codifican de acuerdo a la sintaxis especificada a continuación.

3.5.4. Sintaxis y semántica del campo de datos de un paquete PES de datos de Teletexto

El paquete PES para datos de Teletexto transporta el flujo de datos de Teletexto codificados dentro del campo de datos del paquete PES según se define en la Figura 3.29 en la página siguiente. A continuación se detalla la semántica de sus campos:

- data_identifier: este campo de 8 bits identifica el tipo de datos transportados en el paquete PES. Su valor debe ser el mismo para cada paquete PES que lleve datos del mismo flujo de datos de Teletexto. Sus valores posibles se muestran en la Tabla 3.11 en la página siguiente:

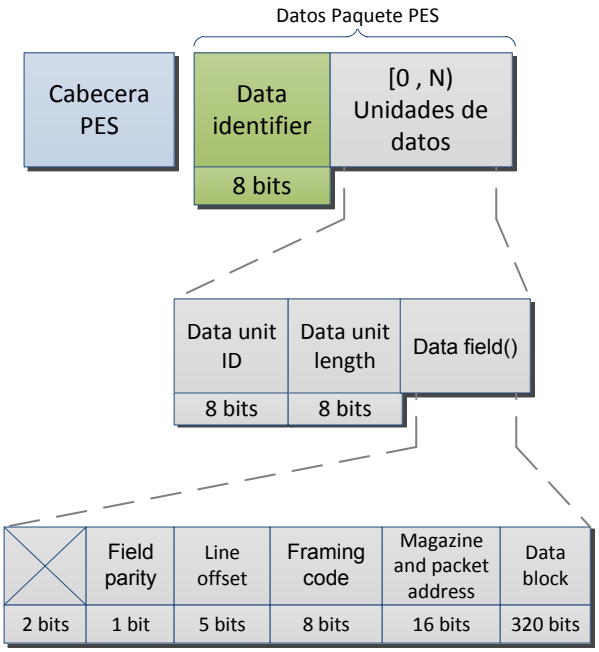


Figura 3.29: Campo de datos de un paquetes PES de datos de Teletexto.

Valor	Descripción
0x00 to 0x0F	Reservado para uso futuro.
0x10 to 0x1F	Datos EBU.
0x02 to 0x7F	Reservado para uso futuro.
0x80 to 0xFF	Definido por el usuario.

Tabla 3.11: Valores posibles del campo data_identifier [17].

- *data_unit_id*: identificador de 8 bits del tipo de unidad de datos. Sus valores posibles son los que están recogidos en la Tabla 3.12 en la página siguiente, aunque para flujos identificados en la información PSI como *DVB Teletext descriptor*, solo están permitidos los valores “0x02”, “0x03” y “0xFF”.

- *data_unit_length*: este campo de 8 bits indica el tamaño en bytes de la unidad de datos a partir de este campo. Para unidades de datos que transporten datos de Teletexto EBU, este campo debe valer siempre “0x2C”.

Valor	Descripción
0x00 to 0x01	Reservado para uso futuro.
0x02	Datos de Teletexto no destinados a subtítulos.
0x03	Datos de Teletexto destinados a subtítulos.
0x04 to 0x7F	Reservado para uso futuro.
0x80 to 0xFE	Definido por el usuario.
0xFF	Unidad de datos de relleno.

Tabla 3.12: Valores posibles del campo data_unit_id [17].

- *field_parity*: indicador de 1 bit que especifica el campo para el cual los datos van destinados; un "1" indica que su destino es el primer campo de un frame, mientras que el valor "0" se asocia al segundo campo de un frame. La alternancia del campo *field_parity* sirve para advertir del comienzo de un nuevo campo.

- *line_offset*: campo de 5 bits que especifica el número de línea en la que los datos de Teletexto deben ser presentados si son transcodificados dentro del VBI. Dentro de un campo, el line_offset debe seguir un orden creciente excepto para el valor "0" que no está definido. El campo line_offset es codificado según como se muestra en la Tabla 3.13, aunque los únicos valores permitidos para subtítulos de Teletexto son el "0x00" y los comprendidos en el rango "0x07" a "0x16".

Valor	Descripción	
	field_parity = 1	field_parity = 0
0x00	Número de línea indefinido.	Número de línea indefinido.
0x01 to 0x06	Reservado para uso futuro.	Reservado para uso futuro.
0x07	Número de línea = 7	Número de línea = 320
0x08	Número de línea = 8	Número de línea = 321
:	:	:
0x16	Número de línea = 22	Número de línea = 335
0x17 to 0x1F	Reservado para uso futuro.	Reservado para uso futuro.

Tabla 3.13: Valores posibles del campo line_offset [17].

- *framing_code, magazine_and_packet_address, data_block*: estos campos se corresponden con los 43 bytes siguientes tal y como se explica en la norma EN 300 706. Los paquetes de datos son insertados en el mismo orden en el que se supone que van a llegar al decodificador de Teletexto o que van a ser transcodificados dentro del VBI. Asimismo, los bits son insertados en el paquete PES en el mismo orden en el que deberían aparecer en el VBI.

Capítulo 4

Herramientas de subtitulado para DVB

4.1. Introducción

A lo largo del capítulo 2 se ha podido apreciar la enorme diversidad de subtítulos que existen en función de la funcionalidad del mismo, del modo de distribución, etc. Esta diversidad de subtítulos ha supuesto la aparición de multitud de herramientas para el manejo y creación de los mismos en los diferentes ámbitos de subtitulado que se pueden dar. Sin embargo, toda esta diversidad, tanto de subtítulos como de herramientas, no se da en DVB, donde el abanico de posibilidades se reduce drásticamente, debido al grado de complejidad que implica la generación de subtítulos en este medio.

Durante la primera parte de este capítulo, se realizará una explicación de tres de las herramientas empleadas para analizar un flujo DVB, mientras que a lo largo de la segunda parte, nos centraremos en poner de manifiesto las principales características de las herramientas de subtitulado y se explicará con mayor grado de detalle, algunas de las suites de subtitulado existentes que son compatibles con el formato de subtítulos DVB.

4.2. Tratamiento de los flujos DVB

A continuación, en esta primera sección del capítulo, se va a realizar un recorrido por las herramientas usadas para analizar y monitorizar un flujo DVB. Representan un conjunto de aplicaciones cuyo fin último no es la generación de subtítulos pero que pueden ser de vital ayuda a lo largo del aprendizaje y comprensión de DVB.

4.2.1. Project-X - DVB demux Tool

Esta herramienta software, cuyas principales funciones son la demultiplexión, edición y reparación de archivos MPEG-2 y TS, constituye una gran ayuda en el análisis de datos de un flujo DVB así como un buen complemento durante los primeros pasos en el aprendizaje de este estándar. Está completamente escrita en lenguaje Java por lo que es totalmente portable [18].

Project-X actúa como un decodificador hardware de un Transport Stream MPEG-2, separando el TS en sus diferentes componentes, conocidos como *elementary streams* (ver capítulo 3 en la página 23) [19]:

- ES de audio y vídeo.
- ES de datos: teletexto, subtítulos, etc.

Entre los formatos soportados se encuentran los siguientes:

- DVB MPEG2 transport stream (DVB MPEG2 TS).
- MPTS (*multiple program transport stream*).
- *Packet Video Audio* (PVA, PSV, PSA, PAV).
- MPEG *Program Streams* (MPEG1/2 PS).
- *Linux Video Disc Recorder* (Linux VDR).
- *Packetized Elementary Stream* (PES RAW).
- Elementary Stream (ES).

Entre otras de sus virtudes está la posibilidad de usarlo mediante línea de comandos, lo que facilitaría su integración en otros proyectos, o a través de la interfaz gráfica. En este caso, es posible obtener información valiosa relativa al TS como los PID's de los diferentes ES, el número de ES que hay de cada tipo, realizar previsualizaciones de los programas de un múltiplex, etc. En la Figura 4.1 en la página siguiente, se puede observar un ejemplo de la GUI de Project-X.

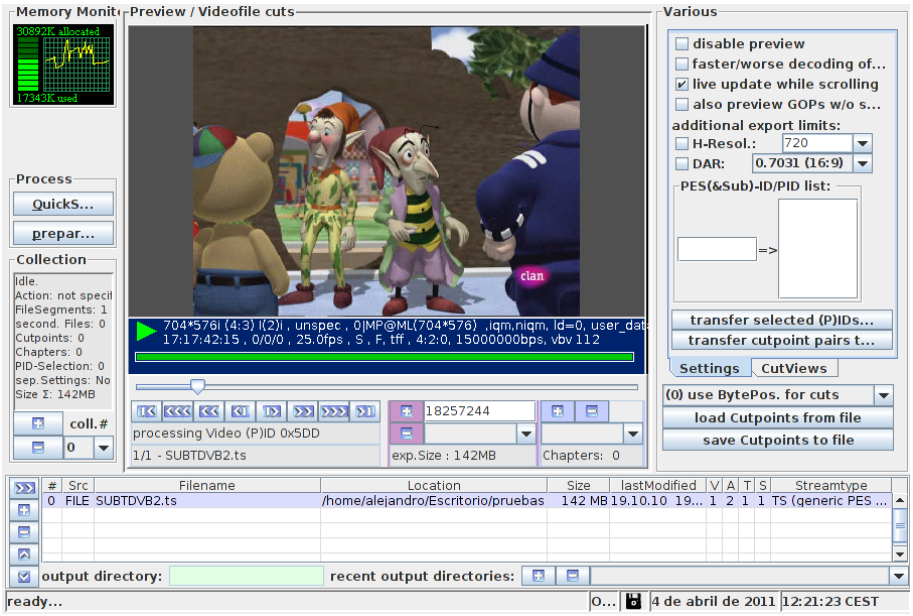


Figura 4.1: Interfaz gráfica de Project-X.

4.2.2. VLC media player

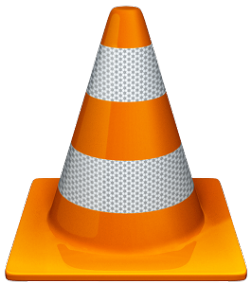


Figura 4.2: Logotipo de VLC media player.

VLC media player (inicialmente conocido como VideoLAN Client[1]) es un reproductor multimedia de código abierto distribuido bajo la licencia GPL[2]. Se inició como un proyecto de estudiantes de la École Centrale Paris aunque actualmente es un proyecto desarrollado a nivel mundial [20].

Es un reproductor portable y multiplataforma, con versiones para los siguientes sistemas operativos:

- Microsoft Windows.
- GNU/Linux.

[1]Ya no tiene sentido esta sigla puesto que actualmente VLC actúa como cliente y servidor. En sus inicios, coexistieron VLC y VLS, la parte servidora.

[2]GNU General Public License. Es una licencia creada por la Free Software Foundation en 1989 (la primera versión), y está orientada principalmente a proteger la libre distribución, modificación y uso de software.

- Mac OS X.
- BeOS.
- FreeBSD.
- Solaris.
- Windows CE.

Una de las principales ventajas de VLC es que incluye de forma nativa un gran número de codecs libres, evitando la necesidad de instalar o calibrar codecs propietarios. Muchos de los codecs incluidos en VLC son proporcionados por la biblioteca libavcodec del proyecto FFmpeg, que será descrito en la subsección 4.2.3.

Disponer de esta variedad de codecs le convierte en uno de los reproductores más versátiles que existen, dado que es capaz de leer una gran cantidad de formatos de audio, vídeo y subtítulos, así como todo tipo de contenedores. Existe una descripción detallada de todos ellos en "*VLC Features Formats*" [21].

Otro aspecto muy cuidado de este reproductor es la cantidad de información que pone a disposición del usuario, lo cual puede resultar muy útil durante el estudio de MPEG-2 y otros formatos, pues ofrece información muy detallada sobre el códec usado en cada caso.

4.2.2.1. Streaming en VLC

Hasta ahora se han mencionado todas las características de VLC que lo diferencian como reproductor. No obstante, VLC pone a nuestra disposición una amplia gama de soluciones de streaming actuando como servidor o como cliente. Entre las posibilidades existentes de streaming se encuentran las siguientes:

- Como servidor (unicast[3] y multicast[4] en IPv4 o IPv6):
  - Archivos MPEG-1, MPEG-2 y MPEG-4 / DivX.
  - DVD's.
  - Desde una tarjeta codificadora MPEG.
  - Desde una cámara DV.
  - Vídeos en directo en la red.
- Como cliente: recibir, decodificar y representar flujos MPEG, ya sean emitidos por la salida de emisión de VLC o por otra aplicación.

La Figura 4.3 describe en esencia, el papel que puede desempeñar VLC a lo largo de la cadena de streaming.

[3]Unicast: a una sola máquina.
[4]Multicast: a un grupo dinámico de máquinas.

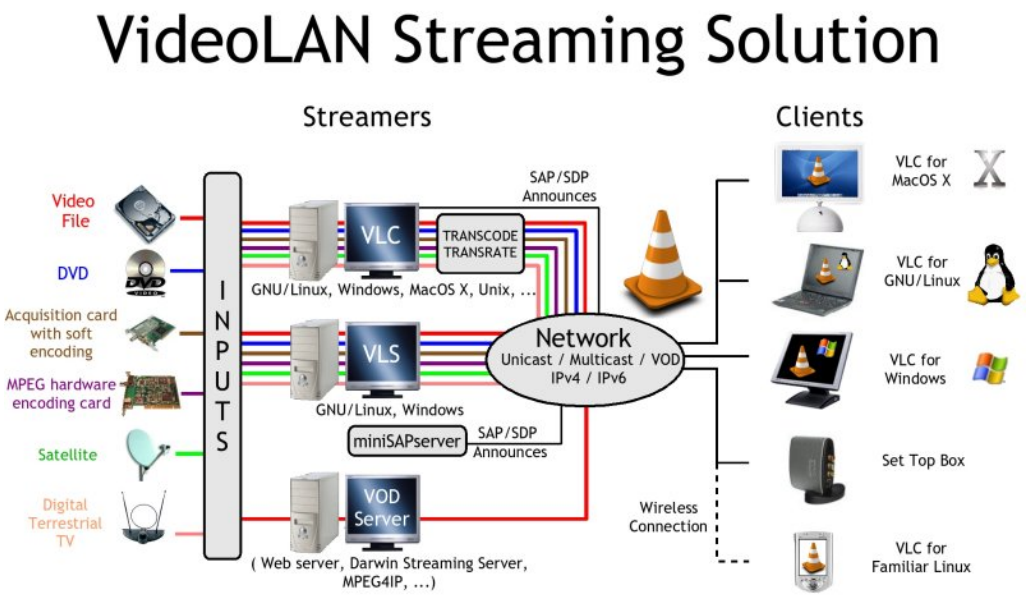


Figura 4.3: Streaming en VLC.

4.2.2.2. VLC por línea de comandos

VLC media player dispone de una potente y compleja línea de comandos desde la que se puede replicar cualquier acción disponible mediante la GUI y extender aún más las posibilidades que ésta ofrece. Explicar todos y cada uno de los comandos disponibles en VLC llevaría mucho tiempo y espacio y, puesto que este no es el propósito del proyecto, simplemente se va hacer referencia a las opciones interesantes en lo que al proyecto se refiere. No obstante, si se desea obtener una información más completa sobre los comandos de VLC, se puede encontrar en "*VLC command-line help*" [22].

Al igual que con Project-X, durante el análisis del estado del arte del proyecto, se centró nuestra atención en aquellas herramientas que tuviesen habilitada una línea de comandos mediante la cual pudiéramos invocar las funciones que nos interesasen. La mayor dificultad del proyecto reside en la generación del flujo de subtítulos, lo que nos llevó a buscar entre los comandos, aquellos que nos permitiesen alcanzar tal fin. A pesar de que esto no constituía una solución a nuestro problema, puesto que era necesario que esto se pudiese hacer de forma dinámica a partir de datos de audio, vídeo y subtítulos cambiantes en función del tiempo, permitía explorar las enjundias de la generación de subtítulos y representaba un paso intermedio antes de abordar la creación de los mismos. Para esto propósito, la mejor aproximación encontrada fue a través del siguiente comando:

```
% cvlc video.avi --sout '#transcode{senc=dvbsub}:
std{access=file,mux=ts,dst=prueba.ts}' --spu --sub-file sub.srt
```

En esencia, lo que se conseguía a través de esta secuencia era generar un Transport Stream con subtítulos DVB-SUB (mediante codificación basada en cadenas de caracteres) a partir de un archivo de vídeo AVI y un archivo de subtítulos en formato SRT. Sin embargo, se encontraron fallos en la salida generada, que se pudieron comprobar y modificar haciendo uso de un editor hexadecimal. En cualquier caso, y a pesar de estos errores, este comando nos permitió visualizar un ejemplo de DVB-SUB mediante caracteres y corroborar el significado de los bytes que componen un subtítulo mediante la comparación con la norma explicada en la sección 3.4.

4.2.3. FFmpeg

Figura 4.4: Logotipo de FFmpeg.

FFmpeg es una solución software multiplataforma distribuida bajo licencia *LGPL*[5] o GPL, dependiendo de las opciones de configuración elegidas, creada para grabar, convertir y emitir audio y vídeo. Incluye la librería *libavcodec*, líder en codecs de audio y vídeo, y que actualmente es usada por una gran cantidad de proyectos externos a FFmpeg [23]. Este software comenzó como una iniciativa de Fabrice Bellard y fue desarrollado bajo GNU/Linux aunque puede ser compilado bajo los principales sistemas operativos entre los que están Microsoft Windows o MAC OS X [24].

FFmpeg dispone de una lista extensa de formatos de fichero, ya sea de vídeo, audio, imagen o subtítulos, así como de una enorme variedad de codecs. Esto es posible gracias a las librerías de las que dispone, en especial *libavformat* y libavcodec y a que sus creadores dejan abierta la posibilidad de usar librerías externas para extender aún más su funcionalidad. Esta lista está accesible en "*FFmpeg General Documentation. Supported File Formats and Codecs*" [25] o desde el módulo ffmpeg, mediante las opciones *-formats* y *-codecs*.

El proyecto está compuesto de diversos componentes lo que hace que sea un software muy modular, y de ahí, que sea usado en muchos proyectos de forma externa [26]. Entre los proyectos más conocidos que basan parte de su funcionalidad en alguno de sus componentes encontramos los siguientes ejemplos [27]:

[5] *GNU Lesser General Public License.*

- *Google Chrome*: el navegador web de Google.
- *MPlayer*: reproductor de películas multiplataforma.
- *SUPER*: *Simplified Universal Player Encoder & Renderer*. Potente reproductor y conversor de archivos multimedia.
- VLC media player: reproductor multimedia descrito en la subsección 4.2.2.
- *xine*: reproductor multimedia gratuito.

A continuación se realiza una breve descripción de los principales componentes que constituyen FFmpeg [26].

4.2.3.1. *ffmpeg*

Herramienta en línea de comandos para convertir ficheros de vídeo y audio, flujos de red o capturas de cualquier fuente de vídeo y audio como puede ser la entrada de una tarjeta de TV [28].

Su interfaz de comandos está diseñada para ser lo más intuitiva posible de tal forma que ffmpeg intenta averiguar todos los comandos que se pueden derivar por el contexto. La regla general de la CLI es que las opciones se aplican al siguiente archivo especificado, de tal forma que una misma opción se puede aplicar varias veces a diferentes archivos, ya sean de entrada o de salida.

Además de la posibilidad de convertir entre formatos, también es posible realizar conversiones entre tasas de muestras o modificar el tamaño del vídeo al vuelo.

Cuando se realiza una conversión, por defecto, FFmpeg intenta que ocurra con las menores pérdidas posibles, aplicando los mismos parámetros de audio y vídeo para las salidas que los especificados para las entradas. Son muchas las posibilidades que ofrece este módulo que, además de operar como conversor, permite obtener extensa información del archivo multimedia que se desee, haciendo uso del siguiente comando:

```
ffmpeg -i input.ts
```

La sintaxis genérica de la línea de comandos es como sigue:

```
ffmpeg [[infile options]['-i' infile]] ...

    { [outfile options] outfile } ...
```

No obstante se puede encontrar la documentación completa sobre los comandos posibles en "*FFmpeg Documentation*" [28].

4.2.3.2. *ffserver*

Es un servidor de streaming multimedia para todo aquello que ffmpeg pueda usar como entrada (ficheros, flujos, entrada de la tarjeta de TV, cámara web, etc). Para aquellas entradas que produzcan contenido en directo se puede aplicar un retardo a la señal realizando *time shift* sobre la misma. La cantidad de tiempo que se va a poder retardar la señal dependerá del tamaño de almacenamiento fijado en el archivo de configuración ffserver.conf [29].

El streaming se puede realizar sobre los protocolos RTP, RTSP y HTTP. El modo de funcionamiento es sencillo: una instancia de ffserver permanece a la escucha en un puerto especificado en el archivo de configuración. Por otro lado, una o varias instancias de ffmpeg mandan sus flujos de salida al servidor, los cuales deben estar definidos en la sección *<Feed>* del archivo de configuración. Por último, por cada *feed*, se pueden definir diferentes flujos de salida en diversos formatos, cada uno de ellos especificados en la sección <Stream> del archivo de configuración.

Una vez esté todo configurado, se podrá capturar vídeo y audio en tiempo real desde una tarjeta capturadora y emitirlo por Internet tanto a Windows Media Player como a RealAudio player (con restricciones). A pesar de que teóricamente también se puede hacer streaming desde archivo, tal y como se ha mencionado, según los desarrolladores, actualmente esta característica no funciona correctamente.

Para terminar la descripción de este módulo, la sintaxis de su línea de comandos es algo más reducida que la de ffmpeg. En líneas generales sigue el siguiente modelo:

```
ffserver [options]
```

4.2.3.3. *ffplay*

Es un reproductor de medios muy simple y portable que utiliza las librerías ffmpeg y la librería SDL[6]. Principalmente es usado como banco de pruebas de las diferentes APIs de FFmpeg [30].

Su interfaz de comandos genérica es la siguiente:

```
ffplay [options] 'input_file'
```

4.2.3.4. *ffprobe*

Es un simple analizador de flujos multimedia que muestra la información recopilada de forma que sea fácilmente interpretada tanto por el usuario como por una máquina. Esto se debe a que la salida generada por ffprobe es fácilmente parseable por un filtro textual ya que consta de una o varias secciones del tipo:

[6] *Simple DirectMedia Layer*. http://www.libsdl.org/

```
[SECTION]
key1=val1
...
keyN=valN
[/SECTION]
```

El hecho de poder ser interpretado por una máquina posibilita procesados más sofisticados basándose en datos estadísticos.

Una vez más, se describe la interfaz de comandos genérica:

```
ffprobe [options] [‘input_file’]
```

4.2.3.5. *libavutil*

Es una amplia librería de apoyo que contiene funciones de ayuda a la programación, incluyendo generadores de números aleatorios, estructuras de datos, rutinas matemáticas, etc.

4.2.3.6. libavcodec

Librería que contiene todos los codificadores y decodificadores de audio y vídeo de que consta FFmpeg. Es la principal responsable de que ffmpeg no necesite codecs extra para la codificación o decodificación de los archivos multimedia.

4.2.3.7. *libavformat*

Librería con los multiplexores y demultiplexores para los diferentes contenedores multimedia. Presenta una labor similar a la de libavcodec, de la cual depende, pero limitándose a los contenedores.

4.2.3.8. *libavdevice*

Esta librería contiene dispositivos de entrada y salida para la captura y renderizado de *frameworks* multimedia tales como *Video4Linux*, *Video4Linux2*, *VfW*, o *ALSA*.

4.2.3.9. *libavfilter*

Constituye la actual API de filtrado de FFmpeg. Es el sucedáneo de la anterior librería, *vhooks*, y se inició como proyecto en el *Google Summer of Code*. Su propiedad más valiosa es que los filtros pueden presentar múltiples entradas a la vez que diversas salidas [31].

4.2.3.10. *libswscale*

Librería encargada de realizar el escalado óptimo de imágenes, conversiones de espacio de color u operaciones relativas al formato de los píxeles.

4.3. Software de subtitulado

Actualmente, la oferta de software relacionado con la creación de subtítulos es enormemente amplia, abarcando desde aplicaciones específicas que se centran en tareas concretas, hasta suites completas que desarrollan la labor entera de creación del subtítulo. Muchas de estas aplicaciones están orientadas a contenidos digitales en diferido, sin embargo, teniendo en cuenta la naturaleza de este proyecto, lo realmente interesante es software con capacidad para manejar contenido en directo. Esto representa un inconveniente dada la escasez de estas herramientas o el elevado precio de las mismas [9].

Es importante tener en cuenta que, al igual que otras soluciones software de hoy en día, las aplicaciones para el subtitulado y los formatos que éstas manejan están en continua evolución, por lo que resulta relevante conocer los conceptos y funciones básicas que estos manejan para poder utilizarlos de manera eficiente.

A lo largo de las siguientes líneas, se explicarán las principales características que este tipo de aplicaciones suelen compartir para a continuación presentar algunas de las más destacadas dentro las que están orientadas a producir contenidos DVB.

4.3.1. Gestión de los Subtítulos

Ésta es la parte más importante de estas herramientas y aquella que puede definir factores tan importantes como la velocidad y sencillez con que los subtítulos son generados. Mientras que en algunos programas de edición de vídeo se permite la adición de subtítulos incrustados en la imagen mediante la inserción de campos de texto frame a frame, esta labor es tremendamente tediosa y con un coste temporal muy elevado. Por este motivo, las herramientas diseñadas específicamente para la gestión de subtítulos suelen disponer de una tabla que relaciona los tiempos de inicio y fin del subtítulo y el texto del mismo, lo que facilita la edición de los subtítulos (ver Figura 4.5).

4.3.2. Gestión de vídeo y audio

Durante la generación de subtítulos, en la mayoría de ocasiones estos no son entes independientes sino que son función directa de lo que ocurre o se dice en pantalla. Por este motivo, es muy útil disponer del vídeo, el audio o ambos en pantalla para saber qué subtítulo queremos incluir en cada instante de tiempo. De este modo, lo

#	Inicio	Final	Estilo	Texto
22	0:03:24.39	0:03:27.39	Default	Oye, Eldridge, parece que\Nvamos a necesitar una carga...
23	0:03:27.60	0:03:28.72	Default	Entendido.
24	0:03:28.93	0:03:32.31	Default	Calculo cuatro bloques, eso nos dará\Nunos 10 kg. de potencia total...
25	0:03:32.98	0:03:34.98	Default	Esa explosión va a dirigirse\Nen esa dirección...
26	0:03:35.94	0:03:38.11	Default	La carcasa probablemente volará allí,\Ny la mayor parte de los fragmentos...
27	0:03:38.23	0:03:40.36	Default	...van a dispararse hacia arriba,\Ncon una hermosa forma de paraguas...
28	0:03:40.49	0:03:41.61	Default	Sí.
29	0:03:42.28	0:03:44.82	Default	Tendremos algunas piezas más pequeñas\Ny fragmentos para este lado...
30	0:03:44.95	0:03:46.87	Default	...pero estaremos bien\Nsi nos ponemos detrás del Humvee.

Figura 4.5: Tabla de subtítulos [9].

normal es habilitar un visualizador de vídeo y una gráfica con la forma de onda del sonido cargado.

Ambas ventanas son necesarias para saber en qué instante comienza un subtítulo y en cuál debe terminar. Además, permiten de una manera visual y rápida, conocer qué puntos del contenido tienen, por ejemplo, zonas de silencio, donde no se deben incluir subtítulos.

A parte de incluir los típicos controles de reproducción de audio y vídeo, algunos programas disponen de funcionalidades más avanzadas como la detección de saltos de escena o sistemas OCR[7] para la extracción de los subtítulos ya integrados en la fuente de vídeo seleccionada. En la Figura 4.6, se muestra un ejemplo de ventana para la gestión de vídeo.

Figura 4.6: Visionado del vídeo [9].

4.3.3. Gestión del tiempo

La gestión del tiempo es esencial en los programas de subtitulado, puesto que la sincronía entre el subtítulo, el audio y el vídeo debe tratarse con extremo cuidado para asegurar la transmisión de la información de manera eficaz al usuario de

[7]Reconocimiento óptico de caracteres.

subtítulos.

Generalmente, se dispone de controles en el programa para delimitar el principio y fin de los intervalos temporales donde incluir el texto del subtítulo.

Además, es posible encontrar utilidades dentro del programa para gestionar el tiempo de manera global, de modo que se pueda añadir un retardo a un grupo de subtítulos o, directamente, a todo su conjunto.

4.3.4. El estilo del subtítulo

La mayoría de herramientas de subtitulado son bastante flexibles a la hora de modificar diversos parámetros del estilo de los subtítulos. Algunos de estos parámetros son los siguientes:

- Posición de la caja contenedora del subtítulo en la pantalla.
- Color de la caja contenedora del subtítulo.
- Transparencia de la caja contenedora del subtítulo.
- Color de letra del subtítulo.
- Estilo de letra del subtítulo.

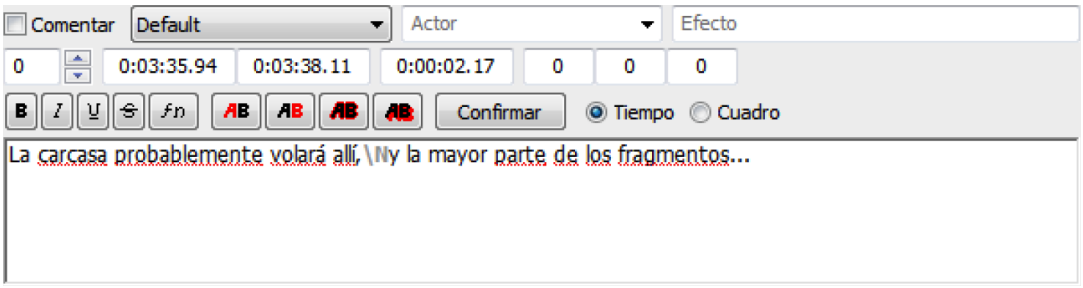


Figura 4.7: Gestión del estilo y texto del subtítulo [9].

4.4. Herramientas de subtitulado en DVB

4.4.1. Softel Swift

Figura 4.8: Logotipo de Softel.

Swift es el nombre que comparten los diferentes programas que componen la oferta de software de la compañía Softel para la gestión completa de los subtítulos desde su creación hasta su distribución. La Figura 4.9 describe la familia de soluciones de las que se compone Swift, clasificándolas en base a las funciones que realiza cada módulo en la cadena de creación del subtítulo.

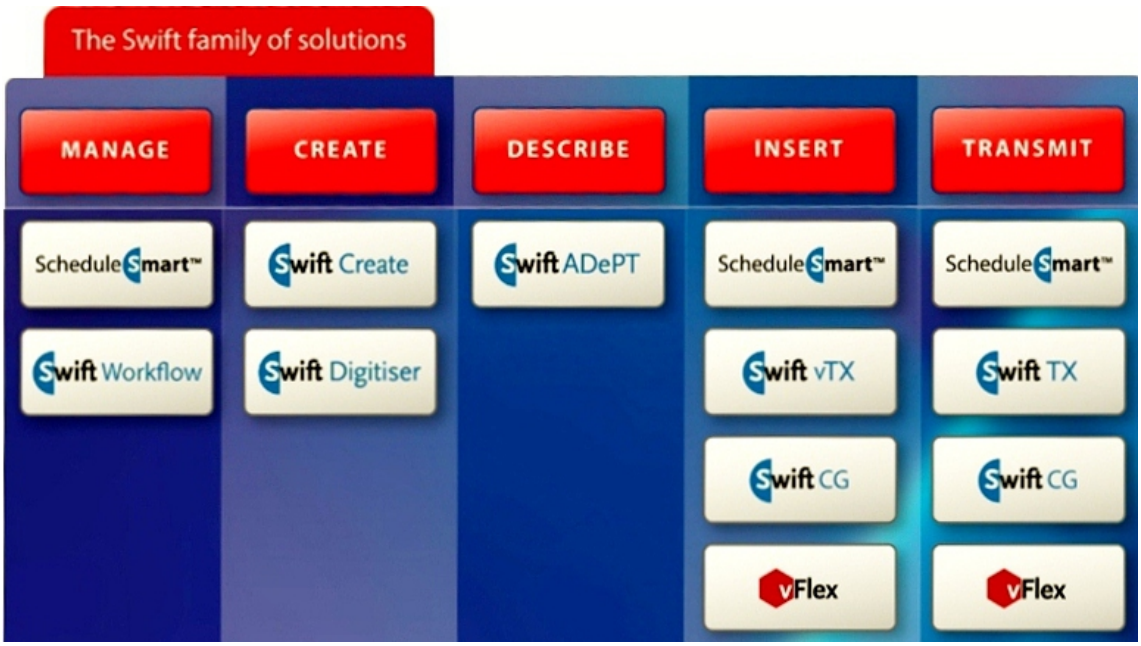


Figura 4.9: Familia de soluciones de Swift [32].

Se trata de una suite de aplicaciones pensada para el ámbito profesional y, pese a su alto coste, de alrededor de 6000€, cuenta entre sus clientes a cadenas como la BBC, RAI, HBO o ESPN [9]. La lista completa de clientes se encuentra en "*Softel subtitling, captioning and ancillary data global customers*" [33].

La flexibilidad de este conjunto de aplicaciones es enorme, siendo capaz de manejar subtítulos en diferido o en directo para ser almacenados o transmitidos a través de la red. Permite así la creación, gestión, transcodificación, inserción y transmisión de los subtítulos en un amplio abanico de formatos, incluidos los subtítulos para sistemas 3D.

Dado el ámbito de este proyecto, nos vamos a centrar en tres de esos componentes: *Softel Swift Create* [34], *Softel Swift TX* [35] y *Softel ScheduleSmart* [36].

4.4.1.1. Softel Swift Create

Tal y como se puede observar en la Figura 4.9 en la página anterior, Swift Create es el módulo encargado de la creación de los subtítulos. Softel proporciona una amplia gama de opciones que extienden Swift Create mejorando su productividad y reduciendo el coste de la preparación y edición de los subtítulos. Estas son algunas de las opciones:

- *Digitizer*: creación de subtítulos a partir de clips de vídeo en formatos no digitales.
- *Video Importer*:procesado de subtítulos a partir de clips de vídeo almacenados en disco.
- *Multi-User*: facilita el subtitulado de contenidos en directo.
- *Subtitle Workflow Manager*: monitorización de las tareas de subtitulado desde la creación hasta la transmisión.
- *Subtitle Translator*: facilita la traducción de los subtítulos.
- *Quality Control*: certifica el trabajo del subtitulador.

Una vez se ha creado el subtítulo, Swift Create puede lanzar Swift TX de manera automática para insertar los subtítulos en el vídeo.

4.4.1.2. Softel Swift TX

Este software es el encargado de la transmisión de los subtítulos y de su unión al contenido, de tal forma que pueda ser presentado al telespectador. La unión entre el subtítulo y el vídeo puede ocurrir en tres instantes de tiempo diferentes definidos por Softel como: *early binding*, *late binding* y *live binding*. Para subtitulado en directo, la opción interesante es la última de ellas. Como se verá en la sección 4.4.1.3, la unión entre subtítulo y contenidos tiene su base en la acción conjunta de Swift TX y ScheduleSmart.

La transmisión de los subtítulos se puede realizar en multitud de formatos aceptando incluso la realización de transcodificación en tiempo real o el recálculo de

tiempos. La Figura 4.10 ayuda a entender el modo en que Swift Create y Swift TX se coordinan durante la creación y transmisión de los subtítulos.

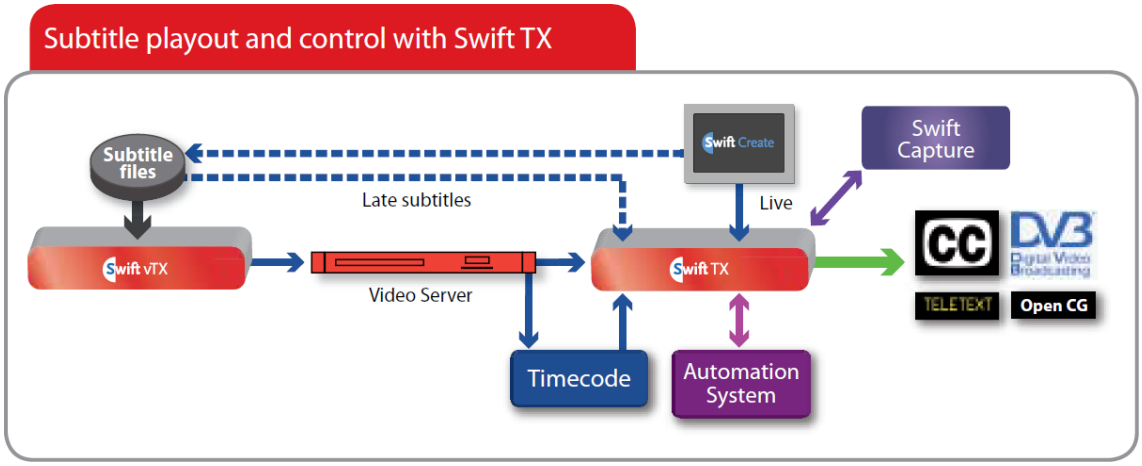


Figura 4.10: Esquema de funcionamiento de Swift TX [35].

4.4.1.3. Softel ScheduleSmart™

ScheduleSmart se combina con Swift TX identificando el punto óptimo de unión entre los subtítulos y el vídeo. Como ya se ha mencionado, esta unión puede ocurrir en los siguientes instantes de tiempo:

- *Early binding*: los subtítulos se insertan en el vídeo mucho antes de la transmisión del contenido.
- *Late binding*: la inserción de los subtítulos tiene lugar casi en directo. Esto es posible gracias a técnicas de codificación más rápidas que el tiempo real.
- *Live binding*: para contenidos en tiempo real como emisión de deportes o noticias.

Así pues, ScheduleSmart se define como el centro de control que determina cómo y cuándo los datos de subtítulos deben insertarse en el vídeo basándose en la información temporal del evento. Estas decisiones se producen sin intervención del usuario o casi sin intervención, permitiendo en la mayoría de casos, un funcionamiento totalmente automático. Una vez tomada la decisión, si estamos en directo o casi directo, Swift TX realiza la transmisión del subtítulo. En caso contrario, el encargado de realizar la inserción de los subtítulos es Swift vTX[8].

[8]No se explica este módulo ya que no es válido para subtitulado en directo.

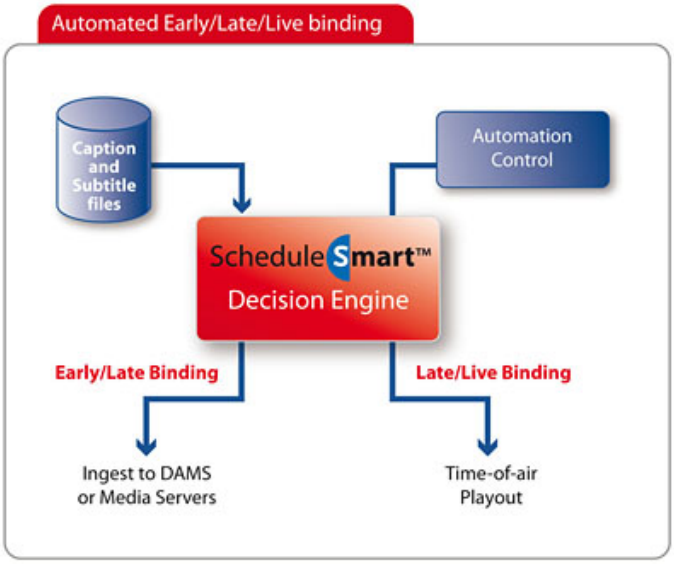


Figura 4.11: Esquema de funcionamiento de ScheduleSmart [36].

4.4.2. FAB Subtitler Live Edition

Figura 4.12: Logotipo de FAB.

FAB Subtitler Live Edition, perteneciente a la familia de programas de la empresa FAB, representa una solución para la transmisión de subtítulos de programas en los que los subtítulos no pueden ser preparados con antelación [37].

Este software es ampliamente utilizado a nivel profesional para la gestión de subtítulos en diferentes cadenas de televisión, muchas de ellas en territorio español, como TVE, Antena 3, Canal+ o Tele 5 [9]. Presenta las siguientes características especiales respecto a la versión estándar del editor:

- Interfaz a sistemas de noticias como ENPS[9], iNews[10], Dalet/ANN[11], etc.
- Teclado estenográfico para entrada rápida de texto.

[9]The essential news production system. http://www.enps.com/
[10]Centro Dinámico de Creación y Distribución de Noticias. http://www.avid.com/es/products/inews
[11]http://www.dalet.com/

- Interfaz al software de reconocimiento de voz de IBM Via Voice o Dragon Naturally Speaking.
- Editor de texto especial para la creación y formateo de subtítulos de la forma más rápida posible.
- Editor de texto especial en el que se puede introducir texto y que éste sea transmitido al instante, autodeterminando su duración.
- Múltiples estaciones de trabajo pueden trabajar en el mismo contenido a la vez.

Debido a estas propiedades, FAB Subtitler Live Edition está especialmente recomendado para emisiones en directo con subtítulos en modo teletexto o DVB, motivo por el que numerosos canales de televisión utilizan este software habitualmente en sus emisiones. Este programa está desarrollado para ser ejecutado sobre el sistema operativo Windows.

4.4.3. Cavena

Figura 4.13: Logotipo de Cavena.

La empresa Cavena, al igual que Softel, divide la cadena de preparación del subtítulo entre diferentes productos, de forma que cada uno de ellos presenta una funcionalidad claramente diferenciada. Una vez más, centramos nuestra atención en aquellos que se enmarcan dentro del ámbito del proyecto, por lo que el *workflow* del subtitulado quedaría de la siguiente manera:

1. *Cavena Tempo*: sistema de preparación de subtítulos [38].
2. *Cavena STU*: codificador y renderizador de subtítulos [39].
3. *Cavena STC*: núcleo del sistema de subtitulado [40].

El software de Cavena está desarrollado para ser ejecutado en sistemas operativos Windows.

4.4.3.1. Cavena Tempo - *Subtitle Preparation System*

Cavena Tempo es un software para la generación descentralizada de subtítulos, es decir, permite crear los subtítulos en un terminal para ser enviados a la línea de emisión, donde serán insertados, lo que nos da la posibilidad de generar los subtítulos de manera remota, en un ordenador dedicado a ello [9].

Cavena Tempo dispone de las características más comunes en herramientas de subtitulado, como la gestión de los estilos de los subtítulos, y dispone también de funciones para la comprobación de la corrección de los textos y tiempos de los subtítulos. Además, brinda soporte a todos los idiomas permitidos por Windows.

4.4.3.2. Cavena STU - *Subtitle Transmission Unit*

Este módulo realiza la tarea de codificación y renderización de los subtítulos preparados por Cavena Tempo. Cada STU se encarga del flujo de subtítulos de un único canal de TV y está diseñado para trabajar dentro de un entorno de subtitulado automático, aunque también permite ser controlado manualmente.

Entre los formatos para la transmisión de subtítulos permitidos están tanto DVB como teletexto, además de otros formatos adicionales [41].

4.4.3.3. Cavena STC - *Subtitle Transmission Control*

Constituye el núcleo del sistema de subtitulado, conectando las diferentes partes de las que se compone y controlándolas de manera automática. Un STC es capaz de controlar hasta 24 STUs, es decir, permite el subtitulado de hasta 24 canales de televisión.

4.4.4. SysMedia WinCAPS

Figura 4.14: Logotipo de SysMedia.

La empresa SysMedia dispone de varias aplicaciones para facilitar el proceso de subtitulado, bajo el nombre de WinCAPS. Estas aplicaciones permiten tanto el subtitulado de archivos multimedia como el de sistemas en tiempo real para difusión. Al igual que la mayoría de suites de subtitulado que hemos mencionado, WinCaps puede trabajar en cualquier idioma y soporta entre los diferentes formatos de subtítulos, el formato DVB.

Sysmedia dispone de innovadoras soluciones para crear los subtítulos en un menor tiempo. Por un lado, dispone de una característica denominada *AutoTime*, que sincroniza automáticamente el texto del subtítulo con el audio correspondiente, de modo que el proceso de introducción de tiempos se agiliza claramente. Por otra parte, mediante *SpeakTITLE*, se permite la utilización de software de reconocimiento de voz para, mediante rehablado, generar más rápidamente el subtitulado [42].

Asimismo, WinCAPS es capaz de realizar procesado y análisis del vídeo para detectar cambios de escena y facilitar el proceso de subtitulado. Este tipo de análisis avanzados son también aplicados al audio del archivo multimedia y a los subtítulos generados, para ayudar al subtitulador en la gestión del corte y formateado general del texto de los subtítulos [9].

Por último, WinCAPS Live permite el subtitulado de programas en directo habilitando la entrada de voz (a través de SpeakTITLE), la entrada de texto mediante teclados estenográficos (módulo *StenoClient*), o software de escritura externo. De forma más concreta, para subtitulado en directo de noticias, WinCAPS News constituye una interfaz a los principales sistemas de noticias como iNews o ENPS.

4.5. Conclusiones

Los mostrados aquí son sólo algunos ejemplos de herramientas de subtitulado disponibles en el mercado. Como se ha podido observar, salvo las herramientas para el tratamiento de flujos DVB, el software de subtitulado para contenidos DVB está muy limitado en lo que a sistemas operativos se refiere, de forma que la mayoría de fabricantes solo prestan soporte bajo el sistema operativo Windows.

El factor común y principal obstáculo que ofrecen estas herramientas es el elevado precio que alcanzan estos productos. En general, cubren la mayor parte de las necesidades que un sistema de subtitulado puede requerir, sin embargo, se muestran más inflexibles en algunos aspectos que imposibilitan el uso de estas herramientas como parte de este proyecto. Entre estos aspectos, se encuentra el hecho de que algunas de las suites no dan la posibilidad de generar los subtítulos mediante reconocimiento de voz, mientras que entre las que sí lo permiten, el ASR únicamente constituye una herramienta de ayuda a la edición del subtítulo y no una forma de automatizar al completo el sistema de subtitulado.

Como consecuencia de esto, ante la imposibilidad de hacer uso de alguna de estas herramientas para el propósito que se persigue, conseguir subtítulos de manera automática y sincronizada, surge la imperiosa necesidad de conseguir la codificación de los subtítulos dentro de este contexto. Para ello, se tomó como punto de partida, la librería **libavcodec**, mencionada en la subsección 4.2.3. Puesto que el motivo de esta parte no es centrarse en los trabajos realizados, se hablará con mayor detalle de la misma, durante el desarrollo y explicación de los trabajos realizados.

Capítulo 5

Windows Bitmap. BMP

5.1. Introducción

Windows BMP, también conocido como *Device Independent Bitmap* (DIB) es el formato de imagen nativo en los sistemas operativos de Microsoft Windows. Mediante BMP es posible almacenar imágenes digitales 2D en diferentes profundidades de color: 1, 4, 8, 16, 24 y 32 bits por píxel. No obstante, los tipos de 16 y 32 bits por píxel no son habituales [43].

Opcionalmente, BMP permite aplicar una compresión sin pérdida de calidad mediante una implementación simple de *run-length compression* para profundidades de color de 4 y 8 bits por píxel. Es un tipo de compresión muy sencilla mediante la cual los datos son guardados de tal forma que los valores repetidos de un color son reemplazados por la cuenta de colores repetidos de forma consecutiva. Esta compresión únicamente tiene sentido cuando la imagen está compuesta de grandes bloques de píxeles con idénticos colores (de no ser así, es posible obtener un archivo más grande que el no comprimido, es decir, compresión negativa), por lo que es una técnica bastante limitada, que rara vez es usada[1]. Además de la compresión, en el formato BMP se contempla la posibilidad de habilitar un canal alfa (cana alfa = canal de transparencia) y perfiles de color.

En cuanto a la ordenación de los datos en Windows BMP, éstos son almacenados en base a múltiplos enteros de bytes empezando por el byte menos significativo.

La razón de que este capítulo se incluya en esta parte radica en que es el formato que se usa durante la generación de los subtítulos (ver sección 10.2), por lo que resulta de vital importancia entender la estructura y el funcionamiento de este tipo de archivos.

[1]En DVB-SUB también se usa *run-length encoding*. Se realizará una explicación más detallada del algoritmo en la sección 3.4 en la página 50, ya que durante la generación de imágenes en formato BMP no se va a dar uso de esta compresión.

5.2. Estructura de BMP

La estructura de un archivo BMP es muy sencilla. Dicha estructura se puede observar en la Figura 5.1. Un archivo BMP se puede entender como una cabecera BMP seguida del DIB propiamente dicho [44].

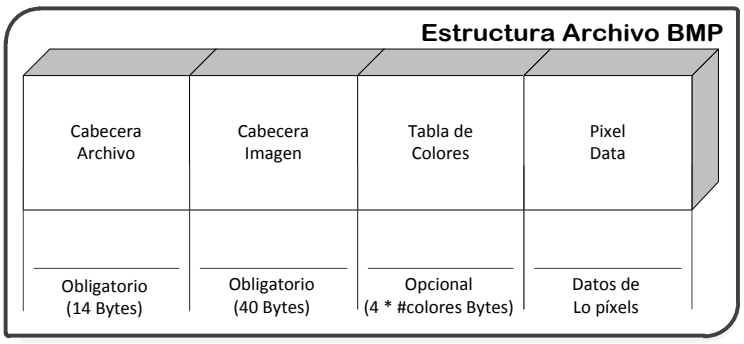


Figura 5.1: Estructura de un archivo Windows BMP.

A continuación realizamos un análisis más detallado de cada una de las partes que componen un BMP.

5.2.1. Cabecera de archivo

Todo archivo Windows BMP comienza con una estructura *BITMAPFILEHEADER* cuya distribución se puede observar en la Tabla 5.1. La principal función de esta estructura es servir como firma que identifica a este formato de archivo.

Nombre del campo	Tamaño en Bytes	Descripción
bfType	2	Contiene los caracteres “BM” que identifican el tipo de archivo.
bfSize	4	Tamaño del archivo en bytes.
bfReserved1	2	Sin usar. Fijados a 0.
bfReserved2	2	Sin usar. Fijados a 0.
bfOffBits	4	Offset al inicio de los datos.
Total	14	

Tabla 5.1: Estructura BITMAPFILEHEADER[43].

Durante el proceso de decodificación, uno de los campos más importantes es el bfOffBits ya que nos permite determinar la posición exacta en la cual los datos de

los píxeles comienzan.

5.2.2. Cabecera de Imagen

La cabecera de la imagen sigue inmediatamente a la cabecera del archivo y puede ser encontrada en dos formas: definida como *BITMAPINFOHEADER* y como *BITMAPCOREHEADER*. La segunda de ellas representa el formato OS/2 BMP mientras que la primera es el formato más común. Por esta razón y dado que es el tipo de cabecera usada en este proyecto, se va a explicar únicamente la cabecera BITMAPINFOHEADER, cuyo tamaño mínimo es de 40 bytes. La distribución de esta estructura es detallada en la Tabla 5.2. A continuación se detallan las características principales de la imagen que esta cabecera determina:

- **Dimensiones de la imagen**: el ancho siempre es un valor positivo mientras que el alto puede ser negativo. Esto se debe a que los píxeles están ordenados empezando desde la última línea de la imagen, es decir, *el primer píxel es el que se encuentra en la esquina inferior izquierda*. En caso de que se quieran ordenar los píxeles empezando desde la primera línea, el valor de la altura se indica en negativo, en cuyo caso no se podrá aplicar ningún tipo de compresión.
- **Resolución o profundidad de color** (*bit depth*): los tipos posibles de resolución son: 1, 4, 8, 16, 24 y 32 bits por píxel.
- **Compresión:** hay tres tipos de compresión permitidos en BITMAPINFOHEADER: RGB (sin compresión), 4-bit RLE y 8-bit RLE.
- **Tamaño del píxel data:** a diferencia de bfSize, este campo determina el tamaño del bitmap propiamente dicho. La forma de realizar el cálculo del mismo es la siguiente.

$$biSizeImage = \left\lceil \frac{biBitCount \cdot biWidth}{32} \right\rceil \cdot 4 \cdot biHeight\, Bytes$$

Tabla 5.2: Estructura BITMAPINFOHEADER[43].

Nombre del campo	Tamaño en Bytes	Descripción
biSize	4	Tamaño de la cabecera. Al menos 40 bytes.
biWidth	4	Ancho de la imagen. Expresado en píxeles.

Nombre del campo	Tamaño en Bytes	Descripción
biHeight	4	Alto de la imagen. Puede ser negativo. Expresado en píxeles.
biPlanes	2	Número de planos. Siempre es 1.
biBitCount	2	Resolución de color del DIB. 1, 4, 8, 16, 24 o 32 bits/píxel.
biCompression	4	Tipo de compresión: 0 = BI_RGB 1 = BI_RLE8 2 = BI_RLE4
biSizeImage	4	Tamaño del bitmap en bytes. Puede ser 0.
biXPelsPerMeter	4	Resolución Horizontal: píxeles/metro.
biYPelsPerMeter	4	Resolución Vertical: píxeles/metro.
biClrUsed	4	Número de entradas usadas de la tabla de color. Si es 0, se obtiene como: $2^{biBitCount}$.
biClrImportant	4	Número de colores importantes.
TOTAL	40	

5.2.3. Tabla de color

Existen tres formatos para especificar la tabla de colores. De los tres existentes, solo se va a explicar el basado en la estructura *RGBQUAD*, por ser el usado en la implementación[2].

Teniendo en cuenta el formato de cabecera de imagen explicado en la subsección 5.2.2, la tabla de color solo aparece para los casos en que $biBitCount <= 8$. Para archivos BMP en formato Windows, la paleta de color consiste en un array de $2^{biBitCount}$ estructuras RGBQUAD para el caso en que $biClrUsed = 0$. En cambio, si biClrUsed es diferente de 0, la paleta de color estará formada por biClrUsed estructuras RGBQUAD.

La Figura 5.2 en la página siguiente constituye un ejemplo del contenido que puede presentar una tabla de color y la correspondiente paleta de colores que se tendría según esos datos.

[2]Se puede encontrar información más detallada en la página 26 de [43].

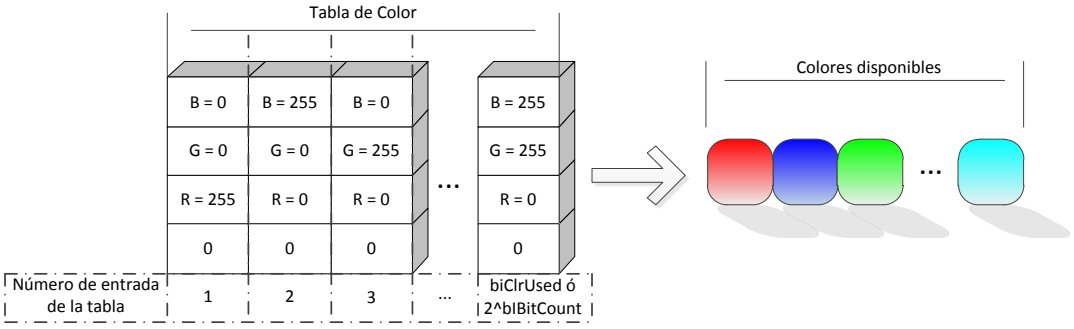


Figura 5.2: Ejemplo de Tabla de Color de un archivo BMP.

Los campos de la estructura RGBQUAD se especifican en la Tabla 5.3[3].

Nombre del campo	Tamaño en Bytes	Descripción
rgbBlue	1	Valor del color azul
rgbGreen	1	Valor del color verde
rgbRed	1	Valor del color rojo
rgbReserved	1	Reservado. Fijado a 0.
TOTAL	4	

Tabla 5.3: Estructura RGBQUAD[43].

5.2.4. Pixel Data

Representa el último bloque del que se compone un archivo BMP. El píxel Data es un array de DWORDs[4] que describe la imagen píxel a píxel. Tal y como se mencionó en la subsección 5.2.2 en la página 105, esta descripción se realiza partiendo del píxel situado en la esquina inferior izquierda de la imagen, recorriéndola de izquierda a derecha, y fila a fila, desde la última hasta la primera. En el caso de que el tamaño de las filas no sea múltiplo de 4 bytes, es necesario añadir bytes de relleno (*Padding*). En la Figura 5.3 en la página siguiente, se muestra un ejemplo del píxel data de una imagen de 2x2 píxeles y profundidad de color de 24 bits.

[3]Bytes de rojos, verdes y azules están en orden inverso (posición rojo se intercambian con azul) de la convención de Windows.[44]

[4]DWORD = Palabras de 32 bits.

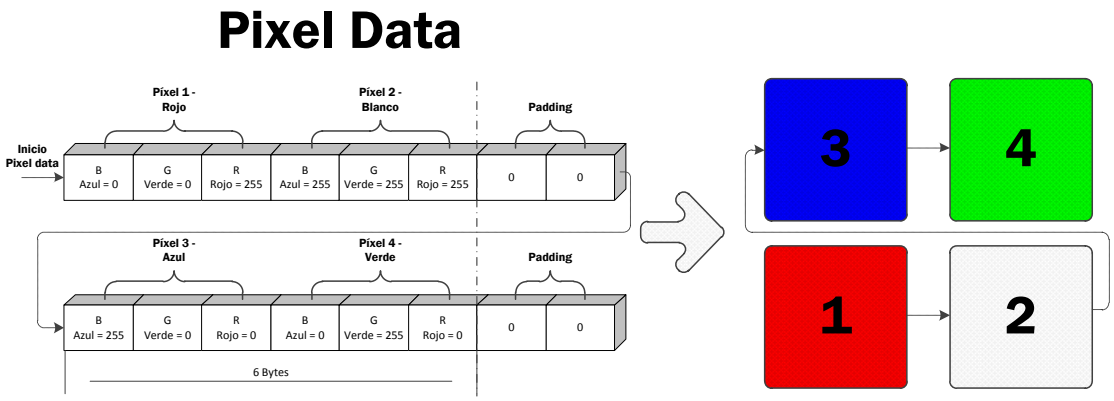


Figura 5.3: Imagen de 2x2 píxeles y profundidad de color de 24 bits.

La figura se compone de dos partes. Por un lado, en la parte izquierda, se muestra el Pixel Data tal y como se encontraría en una imagen BMP, pero para facilitar su comprensión, separado por filas. Dado que la profundidad de color seleccionada es $24\,bits/píxel$ ($3\,bytes/píxel$), una fila de 2 píxeles quedaría representada por 6 bytes:

$$6\,bytes = \frac{3\,bytes}{pixel} \cdot 2\,pixels$$

Tal y como se ha mencionado en el párrafo anterior, las filas deben tener un tamaño múltiplo de 4 bytes. Es por esto que se añaden 2 bytes de relleno al final de cada fila. Además, si nos fijamos en el orden de las componentes de color, éstas están en orden inverso a RGB, es decir, BGR.

Por otro lado, la parte de la derecha indica la forma y orden en que se construiría el bitmap de acuerdo a los datos del píxel Data.

En el anexo F en la página 243 se puede encontrar una tabla resumen de la estructura de un archivo BMP.

Parte III

Trabajos anteriores

Capítulo 6

Apeinta. Apuesta por la enseñanza inclusiva

Figura 6.1: Logotipo de APEINTA.

El proyecto APEINTA: Apuesta por la Educación Inclusiva Dentro y Fuera del Aula propone el uso de nuevas tecnologías tanto informáticas como telemáticas para evitar barreras en el acceso a la educación y de aprendizaje que desafortunadamente actualmente existen en las aulas para todos los estudiantes, independientemente de si estos presentan o no algún tipo de discapacidad [45].

La experiencia adquirida durante este proyecto en el uso de mecanismos de reconocimiento automático del habla así como los resultados obtenidos, han sentado las bases para su utilización en el proyecto recogido en este documento.

6.1. Objetivo

El proyecto APEINTA se centra en dos propuestas inclusivas bien diferenciadas: una de ellas enfocada en su desarrollo dentro del aula y la otra, centrada en la educación inclusiva fuera del aula.

1. Dentro del aula: se utilizan dos mecanismos para tratar de eliminar las barreras de comunicación que aún existen hoy en día en las aulas. En primer lugar, el

uso de mecanismos de reconocimiento automático del habla proporciona una trascripción en tiempo real que será útil para todas aquellas personas que tengan discapacidad auditiva temporal o permanente. Por otro lado, el uso de mecanismos de síntesis de voz (*Text To Speech*, TTS) proporciona apoyo a la comunicación oral entre el profesor y estudiantes.

2. Fuera del aula: se proporciona a los estudiantes una plataforma accesible de enseñanza Web con recursos digitales a los que pueden acceder en todo momento, ya estén dentro de clase como fuera.

El proyecto se centra en facilitar la atención en el aula de personas con discapacidad auditiva, proporcionándoles mecanismos automáticos de subtitulado en directo. Asimismo, también trata de facilitar el acceso fuera del aula a los materiales de estudio y contenidos de la asignatura, proporcionando a las personas con discapacidad auditiva, visual o motora, una aplicación Web accesible que pueden utilizar desde cualquier ordenador, siempre y cuando cuenten con acceso a Internet, donde pueden encontrar todos los recursos del curso de forma accesible, así como vídeos subtitulados y transcritos en varios formatos (evitando de esta manera que las nuevas tecnologías se conviertan en una barrera más de accesibilidad a los contenidos).

6.2. Servicio de Subtitulado

Cuando una persona con discapacidad auditiva , por ejemplo, trata de atender en el aula a la explicación del profesor o a las preguntas que pueden plantear sus compañeros, podría perder información relevante (total o parcialmente) o incluso malinterpretar la información debido a la ausencia de otras vías de comunicación profesor-alumno. Por ello, es necesario proporcionar mecanismos alternativos de acceso a ese contenido auditivo, teniendo en cuenta diversos aspectos y criterios tecnológicos, sociales y pedagógicos para el apoyo a la comunicación oral en el ámbito educativo.

APEINTA proporciona un servicio de subtitulado en directo en el aula, que permite que personas con discapacidad auditiva puedan entender todo lo que se explica y discute en clase.

Así mismo, este servicio puede ser utilizado en directo o diferido a través de Internet para, por ejemplo, subtitular un vídeo que no es aún accesible.

La arquitectura del servicio se muestra en la Figura 6.2, donde se puede observar cómo el sistema capta la voz del profesor, de los estudiantes que quisieran participar en la clase o la señal de audio que se desea subtitular, digitaliza la voz y la convierte en subtítulos o transcripciones que los estudiantes con problemas de audición, por ejemplo, podrían leer o ver en diferentes dispositivos personales como una agenda electrónica (PDA), un ordenador, un teléfono móvil, un tablet PC, un eReader o unas

Figura 6.2: Arquitectura del servicio ofrecido por APEINTA [45].

gafas de subtitulado, así como mostrarlos directamente en un sistema de subtitulado en abierto, como una pantalla de proyección en el aula.

Es importante resaltar el hecho de que los subtítulos se convierten directamente en un material muy valioso para todos los alumnos, ya que se trata de una transcripción directa de todo lo comentado en clase. De esta forma, publicándolos en la plataforma de enseñanza, los alumnos dispondrán de material adicional para complementar los apuntes de clase.

En la Figura 6.3 se muestra un ejemplo donde en el Smartphone de un alumno se recibe la transcripción del discurso del profesor.

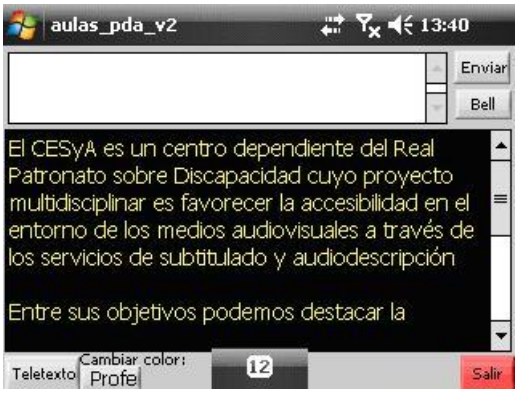


Figura 6.3: Ejemplo de transcripción recibida en PDA [45].

Por otro lado, en la Figura 6.4 en la página siguiente se muestra lo que verían los estudiantes de una asignatura en una pantalla de proyección en un momento concreto de la clase. La imagen del profesor aparece en la parte superior derecha de la pantalla, las transparencias de explicación de la clase aparecen como imagen principal, donde el profesor puede realizar anotaciones según se realiza la explicación, y los subtítulos aparecen en la parte inferior de la pantalla.

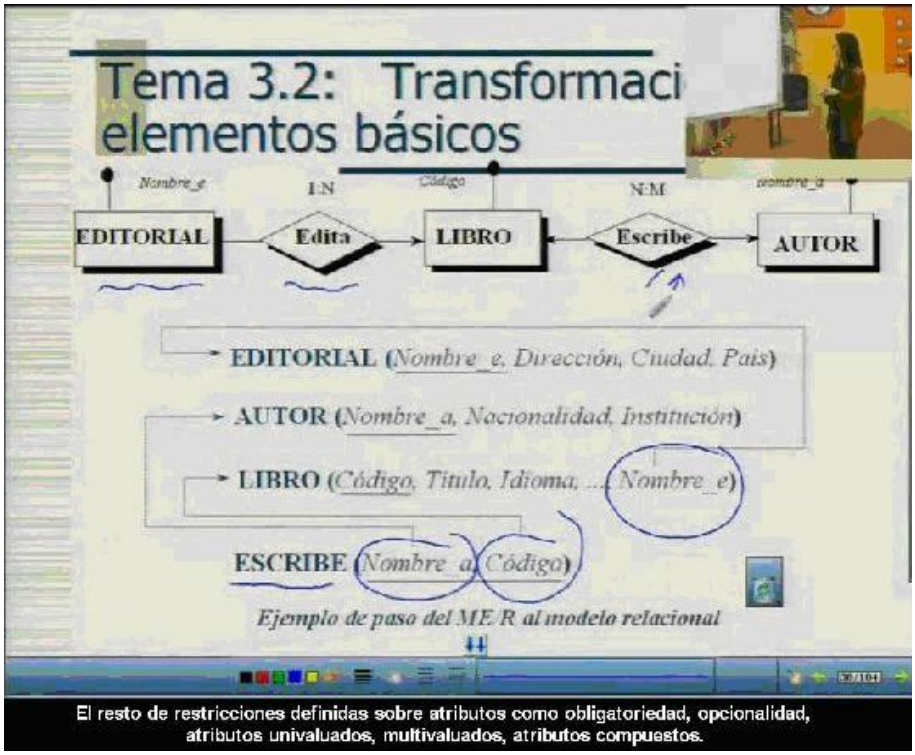


Figura 6.4: Ejemplo de subtítulos mostrados en pantalla [45].

6.3. Servicio de Síntesis de voz

Cuando una persona con algún tipo de problema en el habla o con problemas de expresión en castellano tiene una duda o un comentario en el aula, habitualmente suele tener sus reservas a la hora de participar activamente en la clase con sus comentarios o preguntas. Es por esto que APEINTA también proporciona un servicio de traducción de texto a voz.

Para su utilización, el estudiante escribe en su dispositivo personal (portátil, ordenador personal, agenda electrónica, móvil, tablet PC, etc.) su pregunta o comentario. Éste es enviado inmediatamente a través del servicio de síntesis de voz al servidor, que se encarga de emitir en voz alta y con sonido digital (de ordenador) la pregunta o comentario escrita por el alumno.

6.4. Plataforma de Enseñanza Accesible

APEINTA ofrece, mediante una plataforma de enseñanza digital accesible en Internet, una enseñanza virtual como complemento y/o en combinación a la enseñanza presencial en clase a todos los alumnos. Esta nueva modalidad de aprender y de adquirir contenidos de la asignatura, cuenta con nuevo material docente accesible que se ofrece al alumno en forma de recursos electrónicos multimedia, como vídeos de clases pregrabadas (tanto teóricas como de ejercicios), audio, test de comprobación

de asimilación de contenidos, apuntes electrónicos, etc.

Al ofrecer una plataforma pedagógica en Internet, los estudiantes no necesitan acercarse al aula (evitando las barreras físicas de movilidad) para poder adquirir los conocimientos adecuados de cada asignatura, por lo que la participación del alumno en esta iniciativa puede estar motivada por objetivos como reforzar los contenidos que se han explicado en clase o aprender y adquirir estos conocimientos únicamente por este canal que permite acceder a recursos multimedia. Pero en todo momento, la plataforma digital supone una mejora pedagógica en cuanto a la facilidad de acceso a los diferentes contenidos de las asignaturas para todos los estudiantes, con y sin discapacidad.

Uno de los objetivos principales de la plataforma es el de ofrecer los diferentes recursos educativos en distintos formatos, no sólo para evitar las barreras físicas de acceso al contenido derivado de la diversidad funcional de los estudiantes, sino también para evitar las barreras de acceso y dificultades derivadas del contexto de uso y de las incompatibilidades tecnológicas del dispositivo de acceso empleado (hardware y/o software).

Otro objetivo de esta plataforma de enseñanza es tratar de ayudar a los estudiantes en todo lo posible para adquirir el conocimiento impartido en cada asignatura, por lo que se pueden añadir ejercicios de autoevaluación y recursos pedagógicos que apoyen el aprendizaje de los estudiantes, siempre tratando de adaptar el diseño y recursos de la aplicación al Espacio Europeo de Educación Superior (EEES). Además, la plataforma podrá evaluar el conocimiento de los estudiantes mediante pequeñas pruebas donde se analice si han entendido el contenido de cada uno de los recursos digitales incluidos en la plataforma.

Capítulo 7

Hermes-TDT

7.1. Introducción

El proyecto Hermes-TDT nace de la necesidad de conocer de manera sistemática, automatizada y fidedigna, cuál es el contenido real de la emisión de los radiodifusores que operan en la TDT en lo que se refiere a subtitulado y audiodescripción e información de señalización asociada. El proyecto Hermes-TDT aborda el desarrollo de dos prototipos que permitirán realizar medidas de emisión y de audiencia de los servicios de accesibilidad en la Televisión Digital terrestre. Los prototipos desarrollados en el proyecto permitirán, de manera fiable, automatizada e independiente de los receptores TDT instalados, obtener información actualizada y completa sobre la emisión de TDT [46].

El primer prototipo, denominado *TDT signal sniffer*, permite muestrear la señal de la TDT en una localización geográfica determinada y extraer información relevante acerca de los contenidos que están siendo emitidos en la emisión interceptada, y a partir de esos datos, tipificar exhaustivamente el contenido y la calidad de la señal, especialmente desde el punto de vista de accesibilidad. El segundo prototipo, denominado *TDT audience sniffer*, es un dispositivo que permitirá controlar de forma automática el uso que la audiencia está haciendo de los canales y servicios de accesibilidad de la TDT, así como caracterizar dicha audiencia por perfiles de usuario y conocer sus preferencias.

Hermes-TDT conecta con el proyecto aquí descrito en cuanto a que ofrece la posibilidad de analizar el posible calado que un sistema de este tipo podría tener en la TDT.

7.2. TDT signal sniffer

El TDT signal sniffer es el dispositivo encargado de controlar de forma automática el flujo enviado por los distintos radiodifusores en la señal de la TDT y analizar

que tanto el envío como la señalización, sean realizados de acuerdo a las normas MPEG-2 [14] y DVB-SI [15]. Entre los datos registrados, se pretende recoger información relativa a los servicios de accesibilidad en la TDT: subtitulado de canales, audiodescripción y audio complementario y señalización asociada. La Figura 7.1 muestra esquemáticamente la esencia de la idea.

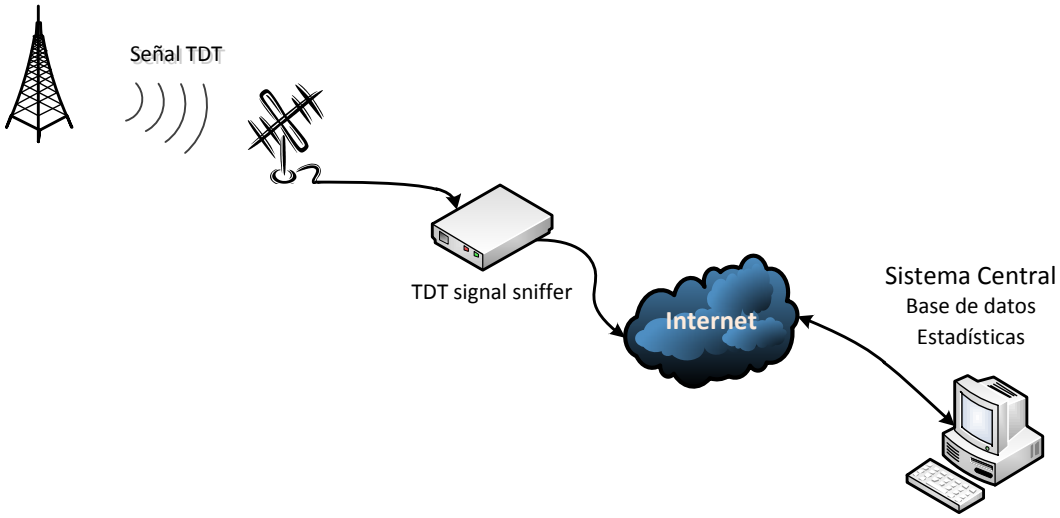


Figura 7.1: TDT signal sniffer [46].

Este sistema permite analizar el flujo de datos de la señal de televisión digital terrestre con los parámetros que se hayan dispuesto (flujo de datos del subtitulado, flujo de datos de la audiodescripción, señalización en pantalla, tipo de programa que posee accesibilidad, horas, etc.), los cuales son inmediatamente almacenados en una base de datos, junto a un barrido temporal de la señal del vídeo como prueba de la veracidad de los datos recogidos. Asimismo, facilita el acceso a los datos para la extracción de estadísticas y análisis.

7.2.1. Funcionalidad

La funcionalidad ofrecida por el TDT signal sniffer comprende:

1. Histórico de eventos televisivos (información histórica actual de la EIT emitida).

2. Histórico de programación avanzada (información histórica avanzada de la EIT emitida).

3. Presencia de subtítulos: detección de la presencia (real) de subtítulos.

4. Extracción de los subtítulos: consiste en separar y almacenar el flujo de subtítulos de un canal. Debido a la diferencia entre la naturaleza de los subtítulos de Teletexto y los DVB-SUB, se definen dos formatos de extracción:

   a) Subtítulos de Teletexto: realizando un volcado del texto (que se transmite como tal) y registrando los códigos de color, posición en la pantalla, etc.

   b) Subtítulos DVB-SUB: realizando una recomposición ("renderización") del mapa de bits, previo al almacenamiento de las imágenes y/o marcas de tiempo.

5. Obtención de texto a partir de los subtítulos: una vez que se ha realizado la extracción de los subtítulos, se puede proceder al reconocimiento del texto mediante técnicas de OCR, a partir de las imágenes reconstruidas del flujo de subtítulos DVB-SUB.

6. Medidas de calidad del subtitulado: en general, consiste en analizar que se cumple la norma UNE 153010[1].

7. Asignación de colores para la identificación de personajes: se comprobará siempre que sea posible, que se sigue la estrategia de asignación de colores para los subtítulos de los distintos personajes según la norma UNE 153010.

8. Señalización correcta: se pretende comprobar que la señalización se hace de forma correcta de acuerdo con la norma.

9. Coherencia señalización – datos: una vez se ha comprobado que la señalización es correcta, se comprobará que realmente se están emitiendo los servicios de accesibilidad que se señalizan.

10. Grabación de muestras: comprende la posibilidad de grabar muestras de cada canal y de su emisión subtitulada.

11. Minutado de subtitulado: pretende medir el minutado de subtitulado por canal en un periodo de tiempo.

7.3. TDT audience sniffer

El TDT audience sniffer es un dispositivo que permite controlar de forma automática el uso que la audiencia está haciendo de los canales y servicios de accesibilidad de la TDT. El equipo permite capturar los datos que caracterizan la audiencia (especialmente la audiencia "accesible") en un hogar determinado, y se conecta a la red

[1]La norma UNE 153010 aspira a homogeneizar convenciones y marcar unos requisitos mínimos de calidad para los subtítulos del teletexto destinados a personas con déficit auditivo.

de datos por medio de un HW TDT vía wifi, Ethernet o módem desde el hogar del usuario, para transmitir los datos de audiencia registrados en su receptor/mando (canal que está en emisión, activación de servicios de accesibilidad, etc.) a una central de información en la que se analizará la audiencia de servicios de accesibilidad en TDT por parte de los usuarios.

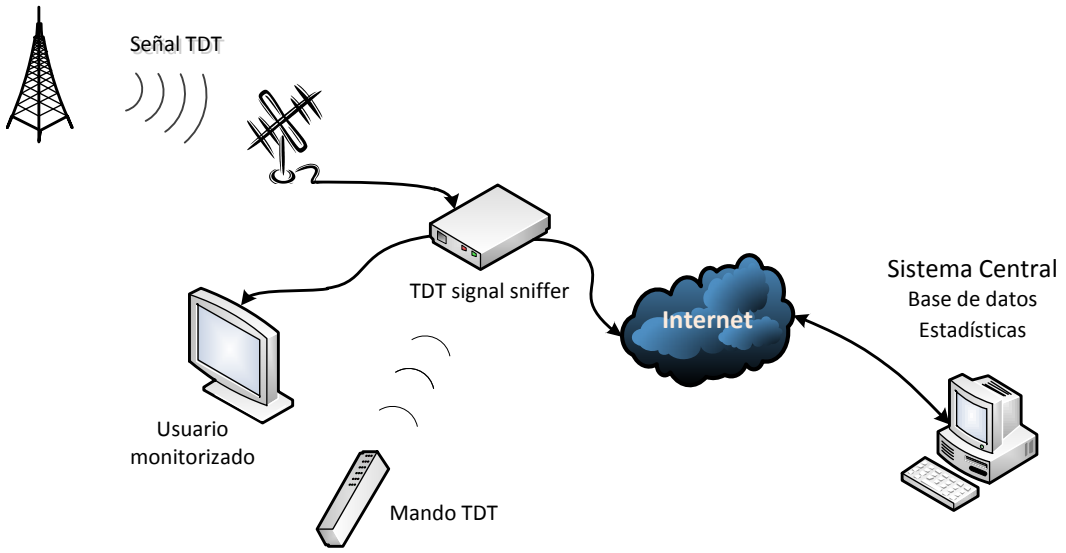


Figura 7.2: TDT audience sniffer [46].

El sistema TDT Audience Sniffer está compuesto principalmente por dos elementos, el sistema base y la aplicación móvil.

- El sistema base es el que actúa como un *Set Top Box* sintonizando los canales TDT, presentando la información y generando los eventos auditables que serán utilizados en las estadísticas.
- La aplicación móvil se descarga desde la base por bluetooth a un terminal móvil. Esta aplicación permite controlar la base de manera remota y editar el perfil del usuario, consultar la guía de programación, etc.

7.3.1. Funcionalidad

Funcionalmente, el sistema debe realizar acciones diferentes tanto en la parte móvil como en el sistema base. Las funcionalidades de la aplicación móvil se describen a continuación:

1. Control remoto: es una herramienta que permite al usuario utilizar su móvil como control remoto. De esta forma, el usuario verá en su pantalla un esquema de opciones asociadas al control remoto y podrá utilizar su teclado numérico para interactuar según las instrucciones en pantalla.

2. Edición del perfil: la aplicación móvil permite editar el perfil del usuario y cargarlo posteriormente en la base. El perfil será definido en función de las categorías y canales favoritos y las ayudas técnicas por canal que desee utilizar el usuario en el momento de visualización del canal.

3. Carga de perfil: una de las opciones de las que dispone el usuario es la carga de perfil. Al pulsar esta opción, el terminal móvil conecta con la base por bluetooth y le envía la información del perfil del usuario en formato XML. Este XML es tratado por la base y presentará a partir de ese momento la información en base al perfil cargado.

4. Consulta de programación: el usuario dispone de la opción de consultar la guía de programación desde su terminal. Para ello, el terminal se conecta a la base mediante una conexión bluetooth y descarga la información más reciente de la EPG. Si esto no es posible, mostrará la última EPG almacenada en el terminal. La información de la EPG es transmitida en XML.

La base del sistema es la encargada de conectar con el flujo DVB y replicar las acciones que el usuario realiza sobre el STB, ya sea a través de su mando de televisión o su terminal móvil. A continuación se describen las funcionalidades del sistema.

1. Visualizar la programación: dependiendo de la elección del usuario, el sistema base visualizará en la televisión el canal que éste desee, así como la información relativa a la programación asociada en ese instante. En caso de que algún canal no pudiera ser sintonizado, el usuario recibirá una notificación en pantalla con el problema en cuestión.

2. Visualizar la EPG: el usuario dispondrá de la posibilidad de acceder a la guía de programación, la cual se mostrará en el televisor del usuario mostrando la información descargada del receptor DVB-T.

3. Activar/Desactivar ayudas: el usuario tendrá activadas o desactivadas en función del perfil cargado, las ayudas técnicas de cada canal. En cualquier caso, el usuario podrá modificar las ayudas en la visualización, así como hacer permanentes los cambios en el perfil activo.

4. Descubrimiento de usuarios: el SW base iniciará automáticamente, y en un periodo configurable de tiempo, la detección de terminales en el entorno. Al descubrir un terminal móvil (que disponga de la aplicación activada), se mostrará un mensaje en el terminal (sólo en aquellos cuyo perfil no esté activo) indicando si desea activar su perfil.

5. Transmisión de estadísticas: cada cierto periodo de tiempo configurable, el sistema base recoge toda la información recopilada de la interacción del usuario

con el sistema y envía las estadísticas al servidor. Antes de realizar el envío, se comprueba la conectividad del sistema con Internet y en caso de múltiples fallos de conexión, se notificaría al usuario para que chequeara si su sistema está correctamente conectado a Internet.

6. Edición del perfil: el usuario podrá editar o eliminar su perfil directamente sobre el sistema base, a través de la interfaz que se le presenta en televisión.

7. Descarga de la aplicación cliente: el sistema ofrece la opción de descargar el software del móvil en un dispositivo. Para ello, el usuario debe acceder a la opción de descarga de software donde se le indicaría al usuario que conecte el bluetooth del terminal en el que desea recibir el software. A continuación, el sistema busca los dispositivos a su alrededor y muestra el listado al usuario, el cual elegiría el terminal donde desea descargar la información. Haciendo uso del protocolo OBEX, la aplicación es transferida al terminal, donde el usuario podría instalar la aplicación siguiendo las instrucciones que se muestren en pantalla.

Parte IV

Descripción de los trabajos realizados

Capítulo 8

Requisitos y Funcionalidad

8.1. Requisitos

Esta sección pretende poner de manifiesto los requisitos que el sistema desarrollado debe cumplir no sólo para poder cumplir con los objetivos descritos en la sección 1.2, sino también para asegurar que pueda ser integrado adecuadamente en el Sistema de subtitulado automático sincronizado. Los requisitos son los siguientes:

Id: UR-001	**Nombre**: Lenguaje de programación
Descripción: El sistema debe ser desarrollado en lenguaje C y bajo el sistema operativo Linux para favorecer la integración en el sistema completo y poder hacer uso de la DVB-API.	

Id: UR-002	**Nombre**: Formato subtítulos
Descripción: Los subtítulos generados deben estar en formato DVB-SUB y dar la opción de elegir entre la codificación basada en píxeles y la basada en caracteres.	

Id: UR-003	**Nombre**: Entrada
Descripción: La entrada del sistema será un Socket TCP por el que se recibirá la información procedente del Subtitle Maker, en base a la cual se van a generar los subtítulos DVB-SUB. La información generada por el Subtitle Maker serán subtítulos ya partidos, con información de colores y referencias temporales referidas al reloj interno del sistema, en el formato que se describe en el requisito UR-004..	

Id: UR-004	**Nombre**: Formato información de entrada
Descripción: El sistema debe ser capaz de interpretar la información recibida por el Socket que se basará en cadenas de texto en un formato XML denominado TTML [47]. La información recibida serán párrafos TTML serializados a los que se añade el atributo color para poder especificar el color sel subtítulo.	

Id: UR-005	**Nombre**: Salida
Descripción: La salida del módulo desarrollado, será una cola de sistema por la que se entregarán los subtítulos construidos al Muxer. Los subtítulos generados deben ser entregados con suficiente antelación de forma que se pueda producir la sincronización.	

Id: UR-006	**Nombre**: Formato información de salida
Descripción: Por la cola de salida circularán paquetes de transporte de 188 bytes cuyo PID será el 7745.	

Id: UR-007	**Nombre**: Sincronización
Descripción: El sistema se hará cargo de que la sincronización de los subtítulos se haga efectiva, realizando los cálculos necesarios para obtener los PTS de los paquetes PES de subtítulos sincronizados. La sincronización se producirá teniendo en cuenta el reloj de referencia de la señal de TDT, la cual va contenida en el PCR de los paquetes de vídeo, y se tomará como punto de partida el PTS del primer paquete de vídeo de la señal que se desea subtitular.	

Id: UR-008	**Nombre**: Temporización
Descripción: El tiempo que el sistema va a tardar en realizar la codificación de cada subtítulo deberá estar acotado y será inferior a 3 segundos. La acción conjunta de todos los módulos que forman el sistema completo no puede retrasarse más allá de 60 segundos. Si esto no sucede, la acumulación de retrasos en cada módulo provocará el descarte de subtítulos en el Muxer del sistema.	

Id: UR-009	**Nombre**: Codificación basada en píxeles
Descripción: La codificación basada en píxeles deberá realizarse mediante CLUTs de 16 entradas y deberá ser reproducible en cualquier decodificador TDT, ya sea hardware o software, como es el caso del reproductor VLC media player. El **color de los subtítulos** generados mediante este tipo de codificación deberá ser alguno de los siguientes: **azul, verde, cyan, rojo, magenta, amarillo, naranja y blanco**. **El tamaño de letra deberá ser 31pt**, dejándose abierta la elección del tipo de letra.	

Id: UR-010	**Nombre**: Codificación basada en cadenas de caracteres
Descripción: La codificación basada en cadenas de caracteres estará orientada únicamente a conseguir los mejores resultados posibles mediante el reproductor VLC media player. Esto se debe a que los decodificadores TDT, en su mayoría no implementan este formato, debido a la necesidad de renderizar las imágenes ellos mismos. Tanto el tipo de letra como el color dependerán exclusivamente del renderizado que el reproductor VLC media player realice. Debido a todo esto, simplemente se pretende demostrar la viabilidad de esta solución a pesar de la dificultad que entraña su implementación.	

Id: UR-011	**Nombre**: Aspectos comunes
Descripción: Ambos tipos de codificación deberán dar lugar a *subtítulos centrados en pantalla y de fondo transparente. Éstos estarán compuestos a lo sumo por 2 líneas.*	

Id: UR-012	**Nombre**: Terminación de la aplicación
Descripción: El sistema debe ser diseñado de tal forma que al terminar la aplicación se liberen todos los recursos utilizados de manera ordenada.	

8.2. Funcionalidad

Una vez vistos los requisitos, a continuación se van a describir las funciones que el sistema debe ser capaz de realizar para poder cumplir con los objetivos del proyecto y ajustarse a los requisitos que acabamos de mencionar.

1. Recepción de la información de entrada en formato TTML mediante un socket TCP actuando como la parte servidora de la comunicación.

2. Extracción de la información contenida en los párrafos TTML mediante un parseo de los mismos.

3. Lectura de los ficheros de configuración PTS y T0 para extraer la información temporal de referencia.

4. Generación de imágenes en formato BMP a partir del texto del subtítulo. Las imágenes deben tener una resolución de color de 24 bits/píxel y carecer de compresión. El texto dentro de la imagen estará centrado y con un tamaño de letra de 31pt.

5. Cálculo de los PTS adecuados para cada subtítulo a partir de la información temporal del subtítulo TTML y del reloj interno de la señal TDT, asegurando así su sincronización respecto al vídeo y al audio.

6. Construcción del flujo de subtítulos mediante la codificación basada en píxeles, a partir de las imágenes generadas y los PTS calculados.

7. Construcción del flujo de subtítulos mediante la codificación basada en caracteres, a partir del texto contenido en los párrafos TTML y los PTS calculados.

8. División de los paquetes PES del flujo de subtítulos en paquetes de transporte.

9. Envío síncrono de los paquetes de transporte con destino al Muxer a través de una cola de sistema.

Capítulo 9

Arquitectura del Sistema

9.1. Introducción

Una vez vistos los requisitos y la funcionalidad del proyecto que se plantea, es necesario situar el conjunto de la aplicación desarrollada dentro del sistema completo, que denominamos en la sección 1.1 en la página 3, como Sistema de subtitulado automático sincronizado, de aquí en adelante, el SSAS. De esta forma, vamos a poder justificar muchas de las decisiones de diseño que se han realizado y que están ligadas de manera inherente a la consecución de los objetivos del SSAS y no sólo a los del proyecto desarrollado.

Con tal propósito, en primer lugar vamos a realizar la descripción del SSAS, haciendo hincapié en los bloques que lo forman y en las características principales de los mismos. Acto seguido, se va a explicar el mecanismo de sincronización de los subtítulos respecto al audio y el vídeo. Esto constituye uno de los factores clave del sistema, por lo que es fundamental entender su funcionamiento, y el modo en que afecta a los bloques que forman el SSAS y en especial, a las partes implicadas del proyecto, denotado de aquí en adelante como *DVB-SUB Generator*. Finalmente, centraremos nuestra atención en el DVB-SUB Generator.

9.2. Arquitectura del Sistema de Subtitulado Automático Sincronizado

En la figura 9.1 se puede observar el diagrama de bloques del sistema así como aspectos relativos al ámbito de cada uno de los bloques o el sistema operativo sobre el que corre cada módulo. A pesar de que la mayor parte está implementada en lenguaje C, bajo el S.O. Linux, el reconocedor de habla debe funcionar sobre Windows™ debido a la escasez de este software en otros sitemas operativos. Las subsecciones siguientes describen estos bloques con mayor grado de detalle.

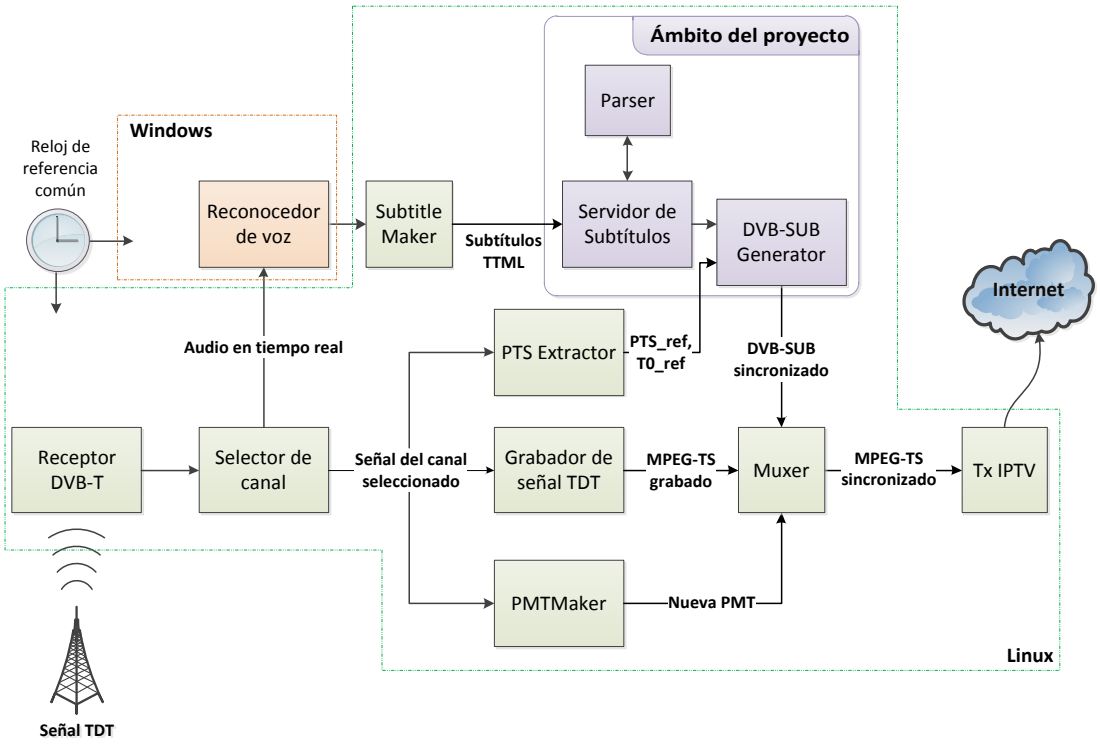


Figura 9.1: Estructura del Sistema de Subtitulado Automático Sincronizado.

9.2.1. Receptor DVB-T

Constituye la entrada del sistema y está formado por la tarjeta sintonizadora de TDT, es decir, este módulo es completamente hardware. Tras realizar un estudio de compatibilidad entre los diferentes modelos existentes en el mercado y el sistema operativo Linux, el modelo de tarjeta escogida fue la *WinTV-NOVA-TD-500 HD* [48]. Sus principales características son las siguientes:

- PCI-Express.
- Doble sintonizador TDT-HD de alta sensibilidad.
- Alta definición HDTV ready.
- MPEG-2/-4.
- Conexiones:
  - Entrada de antena TDT.
  - Entrada de antena TDT para tecnología Diversity[1].
  - Conector del mando a distancia.

[1]Esta tecnología se basa en el uso de dos antenas para generar la mejor señal posible. Es

9.2.2. Selector de canal

El primer paso a realizar por el SSAS es la selección del canal deseado. Para ello, es necesario disponer de la totalidad de canales disponibles en forma de lista, guardados en un archivo de configuración. Esta lista de canales, se puede obtener de forma sencilla mediante una aplicación que hace uso de la DVB-API[2] como es *w-scan*. Una vez ejecutada la aplicación, sus resultados pueden ser volcados a un fichero de texto, en el que se encontrarán los parámetros necesarios de cada canal para ser utilizados a posteriori. Un ejemplo de fichero conteniendo a los canales podría ser:

```
La 2:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_NONE:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:201:203:531
La 1:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_NONE:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:101:103:530
Clan:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_NONE:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1501:1503:533
24h:770000000:INVERSION_AUTO:BANDWIDTH_8_MHZ:FEC_2_3:FEC_NONE:QAM_64:TRANSMISSION_MODE_8K:GUARD_INTERVAL_1_4:HIERARCHY_NONE:1001:1003:532
```

Figura 9.2: Ejemplo de archivo de configuración con lista de canales.

9.2.3. Grabador de la señal TDT (Time shift)

Una vez que el usuario ha seleccionado el canal al que se desea realizar el subtitulado sincronizado (mediante un menú o similar) se procede a la sintonización y grabación de ese canal ayudados nuevamente por la DVB-API. Este proceso se puede realizar mediante un único paso utilizando el programa *dvbstream*. Este programa, permite a la tarjeta sintonizadora anclarse a una determinada frecuencia de la que se han pasado sus parámetros (extraídos del archivo de configuración generado) y almacenar toda la información que recibe de todos los flujos en un fichero con formato transport stream. Este fichero va a contener todos los flujos de vídeo, audio, subtítulos, teletexto y tablas de información (PSI/SI) del múltiplex completo.

Dado que uno de los objetivos es lograr la sincronización de los subtítulos, necesitamos retardar la señal original, lo que implica realizar una grabación del flujo completo de la señal de TDT, para después poder reconstruirla incluyendo los subtítulos sincronizados.

9.2.4. PMT Maker

Según vimos en el Capítulo 3, en la subsección 3.2.8, una parte importante de los flujos de transporte es la tabla PMT, ya que nos permite localizar los elementos

especialmente útil en aquellas situaciones en las que la potencia de la señal TDT recibida no es lo suficientemente alta.

[2]La DVB-API constituye una interfaz entre el sistema operativo y los drivers de una gran variedad de dispositivos receptores de DVB. Representa una gran ayuda durante la sintonización del canal y otras tareas, de ahí la importancia de usar Linux, ya que sólo puede ser utilizada en este sistema operativo. Todas las instrucciones y operaciones realizadas sobre la señal serán realizadas en lenguaje C para acceder con facilidad a las funciones de la DVB-API.

que forman un programa. Por esta razón, si queremos que el flujo de subtítulos sea reconocido por los futuros receptores, no es suficiente con incluirlos dentro del flujo de transporte, sino que además tienen que constar dentro de la PMT asociada al programa que se desea subtitular. Para esto es necesario obtener la tabla PMT del canal que se desea sincronizar y modificarla incluyendo el nuevo flujo de subtítulos.

Tal y como vimos en la subsección 3.2.8.5, la obtención de la tabla PMT de un transport stream no es inmediata, ya que no conocemos directamente su PID, teniendo que extraer este PID de la tabla PAT a la que sí podemos acceder de forma directa.

Para la obtención de las tablas de información y su posterior tratamiento, el sistema hace uso de otra utilidad de la DVB-API: *dvbsnoop*. Esta herramienta permite indicar qué flujo en concreto de la señal deseamos extraer, sean tablas de información, audio, vídeo, etc.

Obtenida la PMT, es necesario modificarla, manteniendo todos los datos presentes y añadiendo la referencia a un nuevo flujo que deberá ser reconocido por el reproductor como un flujo de subtitulado DVB. La modificación, deberá realizarse de acuerdo a los campos definidos en las subsecciones 3.2.8.2 y 3.3.10 a los que habrá que añadir un nuevo stream type junto con el descriptor de subtitulado. **Los datos fijados por el sistema y que deberán ser tenidos en cuenta a la hora de generar el flujo de subtítulos son los siguientes**:

- stream_type: 6.
- elementary_PID: 7745[3].
- descriptor_tag: 89.
    - ISO_639_language_code: spa.
    - subtitling_type: 16.
    - composition_page_id: 1.
    - ancillary_page_id: 2.

Finalmente, la nueva PMT es guardada en un fichero que será utilizado por el Muxer durante el montaje final del transport stream.

9.2.5. PTS extractor

Para la correcta sincronización de los subtítulos, estos tienen que tener una referencia temporal relativa a la reproducción que se está realizando. Esta referencia

[3]En realidad vale cualquier número que sepamos que no va a causar colisión con algún PID de otro canal.

viene indicada por el PCR al que se refieren todos los PTS. El procedimiento utilizado en la aplicación, consiste en la extracción del primer PTS de vídeo encontrado en el archivo del transport stream. De esta forma, se marca un tiempo inicial, tp_{ref}, a partir del cual se pueden sincronizar los subtítulos. La extracción del PTS se realiza comprobando el campo PCR_PID de la PMT que indica el flujo en el que va incluido el PCR (como ya se explicó, en el caso de la TDT suele ser el de vídeo). Una vez se tenga el primer paquete del flujo de vídeo, la consulta se limita a extraer el PTS del paquete, tal y como se encuentra definido en la norma ISO/IEC 13818-1 [14] y como se explicó en capítulos anteriores en la subsección 3.2.3. Este PTS será almacenado en un archivo de configuración para que el DVB-SUB Generator sea capaz de incluir el PTS correcto en cada caso. Adicionalmente, el PTS extractor genera otro archivo de configuración, está vez conteniendo el tiempo de sistema en el que se inicia la grabación del canal seleccionado. La forma en que estos datos van a ser usados por el DVB-SUB Generator se explica en la sección 9.3.

9.2.6. Reconocedor de voz

El reconocedor de voz es el único módulo que funciona sobre el sistema operativo Windows, debido a la escasa disponibilidad de reconocedores de habla en otros sistemas operativos. Como consecuencia de las actividades de investigación en los campos del ASR aplicado al subtitulado de programas de radiodifusión y la sincronización de subtítulos en escenarios seleccionados de televisión en directo, que han sido llevados a cabo en el Centro Español de Subtitulado y Audiodescripción [5], el reconocedor utilizado para el SSAS ha sido el Dragon Naturally Speaking 10.1. La señal de audio de entrada al reconocedor se obtiene directamente de la salida digital de audio del ordenador en Linux, donde se reproduce el canal seleccionado por el usuario. La transcripción del audio junto con los tiempos de sistema asociados a los fragmentos de audio son entregados al *Subtitle Maker*.

9.2.7. Subtitle Maker

El bloque del sistema denominado como Subtitle Maker está escrito en lenguaje Java ya que resulta mucho más fácil manejar cadenas de texto en este lenguaje que en C. La misión del Subtitle Maker es realizar el formateo de las transcripciones y generar los subtítulos de forma que se cumpla la norma UNE 153010. Cada vez que el Subtitle Maker formatea un subtítulo, genera a su salida una cadena de texto muy parecida a la estructura de un párrafo del lenguaje TTML [47], pero que ha sido modificada para que se ajuste a nuestras necesidades. A continuación se muestra la estructura del formato escogido:

- Subtítulo de una línea:

```
<p xml:id="ID subt" begin="Tiempo inicio" end="Tiempo fin" color="color subt">
    Subtítulo
</p>
```

- Subtítulo de dos líneas:

```
<p xml:id="ID subt" begin="Tiempo inicio" end="Tiempo fin" color="color subt">
    Subtítulo 1ª línea<br/>
    Subtítulo 2ª línea
</p>
```

Es importante resaltar que los tiempos que aparecen son siempre de sistema, es decir, tiempos absolutos referidos al reloj interno del sistema.

9.2.8. Servidor de subtítulos

Aunque el Servidor de subtítulos no participa de forma directa en la codificación de los subtítulos, constituye la primera parte del sistema que se encuentra dentro del ámbito que cubre este proyecto. Representa la entrada del DVB-SUB Generator comunicándose con el Subtitle Maker.

Este módulo se encarga de recibir los subtítulos generados por el Subtitle Maker, realizar la llamada al Parser para extraer la información que en ellos está contenida y finalmente, enviarlos al DVB-SUB Generator, incluyendo la información parseada en el siguiente struct:

```
/*
 * Estructura de intercambio para la cola entre el servidor TCP
 * y el DVB-SUB Generator.
 * - mtype: Tipo de msj. Debe ser > 0.
 * - p: El párrafo con el subtítulo y los tiempos
 */
typedef struct msgTTML {
    long mtype;
    Parrafo p;
}TTMLmsg;
```

Tal y como se puede observar, dentro de esta estructura existe otra denominada Parrafo, la cual es explicada en el apartado siguiente. La forma en que se produce la comunicación entre los bloques mencionados se explica a continuación, en la subsección 9.4.1.

9.2.9. Parser

Tal y como se ha explicado en la subsección anterior, el Parser es el encargado de extraer la información útil recibida por el servidor de subtítulos. El código de este bloque se basa en la búsqueda de determinados caracteres que aparecen de forma fija en el subtítulo TTML modificado. Los caracteres objetivo son: <, >, b, /, p, " y el carácter de fin de línea ('\0').

Una vez que se localizan las posiciones de estos caracteres, es posible acceder a la información necesaria para posteriormente construir el subtítulo. Dicha información es almacenada en el struct Parrafo, cuya descripción es la siguiente:

```
/*
 * Estructura donde se almacenan los resultados del Parser
 * - begin: Tiempo de inicio absoluto del subtítulo
 * - end: Tiempo de fin absoluto del subtítulo
 * - num_subs: Número de líneas del subtítulo
 * - sub1: Primera línea del subtítulo
 * - sub2: Segunda línea del subtítulo
 */
struct Parrafo{

    double begin;
    double end;
    char color[COLORSIZE];
    unsigned char num_subs;
    char sub1[SUBSIZE];
    char sub2[SUBSIZE];

}typedef Parrafo;
```

9.2.10. DVB-SUB *Generator*

El DVB-SUB *Generator* es la parte del sistema que se encarga de generar los subtítulos de acuerdo al estándar DVB y que ha sido explicado a lo largo del Capítulo 3. En él reside la mayor parte de la complejidad del proyecto por lo que tras la breve descripción que se va a realizar en este punto, será explicado con mayor detalle en los Capítulos 10 y 11.

Este módulo recibe como entrada las estructuras que genera el servidor de subtítulos a partir de los subtítulos generados por el Subtitle Maker. Dependiendo de la elección del usuario, existen dos posibilidades a la hora de generar el subtítulo DVB-SUB: basado en píxeles o basado en cadenas de caracteres.

9.2.10.1. Codificación basada en píxeles

Este formato es más flexible y agradable en términos visuales que el basado en caracteres, ya que se basa en imágenes que pueden ser diseñadas de acuerdo al criterio que se desee. Internamente, DVB-SUB Generator procesa estas imágenes, generadas en nuestro sistema por el módulo BMP Maker (se describe en la sección 9.4), mediante el algoritmo RLC descrito en 3.4.7.7 en la página 69 y que da lugar a una secuencia de bits que representan a la imagen original, pero comprimida para ahorrar ancho de banda durante la emisión.

9.2.10.2. Codificación basada en cadenas de caracteres

Este formato es mucho más sencillo que el anterior, aunque, por el contrario, menos agradable. En esencia, consiste en insertar la secuencia de caracteres que forman el subtítulo dentro de los segmentos ODS que forman parte del flujo elemental de subtítulos.

Una vez codificada la información del subtítulo en el formato seleccionado, es necesario añadir las marcas de tiempo a los paquetes PES del flujo de subtítulos. A lo largo de la sección 9.3 en la página siguiente, se trata la forma en que estas marcas de tiempo son determinadas.

9.2.11. Muxer

Casi al final del sistema se encuentra el Muxer, que va a ser el dispositivo encargado de construir el flujo DVB de salida, basado en el flujo DVB original y que contiene además los subtítulos perfectamente sincronizados. Durante la multiplexación, las tareas principales son la sustitución de la PMT original por la nueva y la inserción de los paquetes de subtítulos en los puntos adecuados. Para tal efecto, el Muxer compara continuamente los PTS de los paquetes PES de subtítulos con los del vídeo, y en el momento en que su valor es inferior al del siguiente paquetes PES de vídeo que debe ser insertado, introduce el paquete PES de subtítulos. Si nos fijamos en la figura 9.1 en la página 130, las entradas y salidas de este bloque son las siguientes:

9.2.11.1. Entradas del Muxer

- MPEG-TS grabado: es el fichero de la grabación original, obtenido directamente desde la señal de antena y que por lo tanto no contiene los subtítulos DVB sincronizados.
- DVB-SUB sincronizado: son los paquetes de subtítulos correctamente temporizados.

- Nueva PMT: PMT modificada y que incluye la referencia al nuevo flujo de subtítulos.

9.2.11.2. Salidas del Muxer

- MPEG-TS sincronizado: es un flujo similar al de la grabación original, pero en el que ya se encuentran los subtítulos DVB insertados en el flujo correspondiente y sincronizados con el vídeo y el audio originales.

9.2.12. Transmisor IPTV

Como última parte del SSAS se encuentra el módulo encargado de transmitir el flujo de transporte generado vía IPTV. En realidad, este bloque se basa en el software VLC Media Player descrito a lo largo del estado del arte, y de forma más concreta, en su capacidad para realizar streaming. Una vez que el Muxer comienza a generar el flujo de salida, a los pocos segundos el VLC es lanzado como servidor de streaming de forma que el flujo generado pueda ser reproducido de forma remota mediante una instancia del VLC, pero en este caso, como cliente. El protocolo sobre el que se realiza el streaming es sobre RTP.

9.3. Sincronización del subtítulo

Una vez codificada la información del subtítulo en el formato seleccionado, es necesario añadir las marcas de tiempo a los paquetes PES del flujo de subtítulos. Tal y como se ha dicho, los tiempos con los que van marcados los subtítulos generados por el Subtitle Maker, son tiempos absolutos de sistema de la máquina que corre en Windows, los cuales hacen referencia a los tiempos del audio original transcrito. Para conseguir una sincronización adecuada, los relojes de ambas máquinas, la de Windows y la de Linux, deben estar sincronizados. Para ello se hace uso de un servidor de tiempos que proporciona la misma hora a ambas máquinas. Además, hay que hacer referencia al momento en que comienza realmente el programa que se desea subtitular y que vamos a denominar $T0_{ref}$, así como obtener el tiempo relativo de referencia asociado al flujo de transporte que contiene el programa a subtitular, PTS_{ref}, el cual se obtiene a partir del PCR del primer paquete de vídeo.

Ambos datos son extraídos de los archivos de configuración generados por el PTS Extractor.

La figura 9.3 en la página siguiente muestra de forma esquemática toda la temporización necesaria para que los subtítulos pueden ser insertados de forma síncrona.

En primer lugar, a partir de PTS_{ref} es fácil obtener el tiempo de presentación de referencia relativo a la marca temporal tp_{ref}, tal y como se indica en la fórmula 9.1 en la página siguiente:

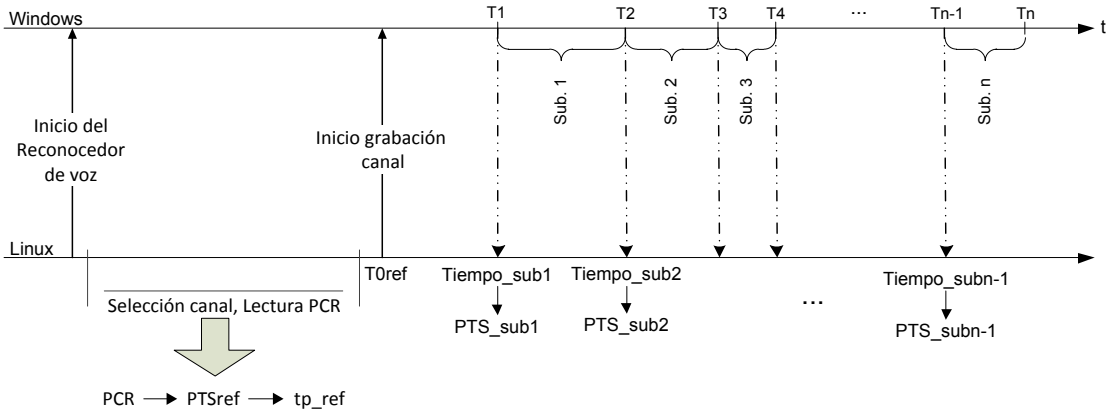


Figura 9.3: Esquema de tiempos para la sincronización.

$$tp_{ref} = {}^{(PTS_{ref} \times 300)}/_{(System\,Clock\,Frequency)} \tag{9.1}$$

Donde System Clock Frequency (S.C.F) es equivalente a $27MHz$.

Una vez que tenemos estos datos, el sistema se encuentra en disposición de insertar los PTS correspondientes a los paquetes PES de subtítulos mediante los dos siguientes cálculos:

$$Tiempo_{sub1} = tp_{ref} + (T_1 - T0_{ref}) \tag{9.2}$$

$$PTS_{sub1} = \left[{}^{(S.C.F \times Tiempo_{sub1})}/_{300}\right] \,\%2^{33} \tag{9.3}$$

Mediante la ecuación 9.2, se obtiene el tiempo de presentación del subtítulo respecto al de referencia, mientras que la ecuación 9.3 permite obtener el PTS que deberá llevar la cabecera del paquete PES del subtítulo. De esta forma, se consigue que los tiempos contenidos en los datos recibidos por el servidor de subtítulos, estén referenciados al PCR del primer paquete de vídeo, y por consiguiente, sincronizados con el audio al que hacen referencia. Es importante mencionar que en todo momento, el valor $(T_1 - T0_{ref})$ debe ser menor que el tiempo que se retarda la señal, de lo contrario, los subtítulos no llegarían a tiempo al Muxer y tendrían que ser descartados.

9.4. Arquitectura del DVB-SUB Generator

Una vez que hemos visto la estructura global del SSAS, podemos centrarnos en el DVB-SUB Generator y en los bloques internos que lo forman, atendiendo a su arquitectura y a las interfaces existentes entre módulos, ya sean de forma interna o

externa.

Tal y como se ha comentado anteriormente, este módulo contempla dos posibilidades a la hora de generar los subtítulos. Así pues, si nos fijamos en la figura 9.4, podemos observar cómo dentro del conjunto denominado como DVB-SUB Generator existen dos opciones dependiendo de la elección del usuario, lo que va a significar que se activen diferentes bloques del sistema en función de dicha elección.

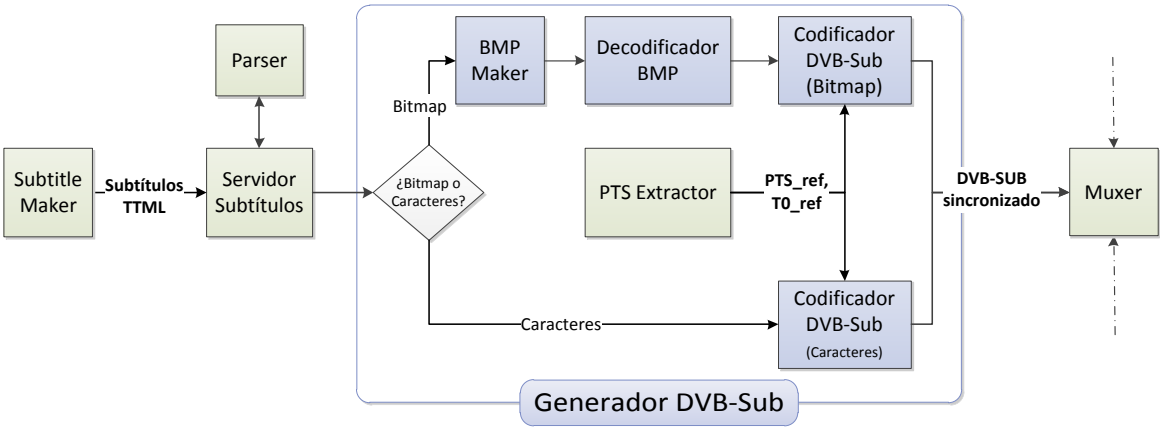


Figura 9.4: Estructura del DVB-SUB Generator.

A continuación se realiza una breve descripción de los bloques internos del DVB-SUB Generator:

- **BMP Maker**: cuando la codificación que se va a seguir es la basada en píxeles, es necesario realizar un procesamiento previo a la codificación del subtítulo. Una parte de este procesamiento es llevada a cabo por el BMP Maker, donde se produce la conversión del texto del subtítulo en imagen. El BMP Maker está escrito en PHP y genera a su salida, un archivo de imagen BMP.

- **Decodificador BMP:** el decodificador BMP es el que interviene en el resto de ese procesamiento previo que hemos mencionado, decodificando la imagen BMP generada por el BMP Maker de forma que el codificador de subtítulos pueda acceder a la información contenida en la imagen en la forma que necesita.

- **Codificador DVB-SUB (Bitmap):** realiza la codificación del subtítulo basándose en la imagen generada por el BMP Maker y tratada por el Decodificador BMP.

- **Codificador DVB-SUB (Caracteres):** realiza la codificación del subtítulo basándose en los caracteres recibidos directamente del servidor de subtítulos, por lo que no necesita el procesamiento previo de la codificación basada en píxeles.

9.4.1. Interfaces

Para terminar con este capítulo, se va a describir la forma en que se comunica el software desarrollado en el proyecto con el exterior. Primeramente, la figura 9.5 recoge gráficamente las interfaces que existen entre el DVB-SUB Generator y el Servidor de subtítulos, con el resto de módulos del sistema, así como la interfaz existente entre ambos módulos.

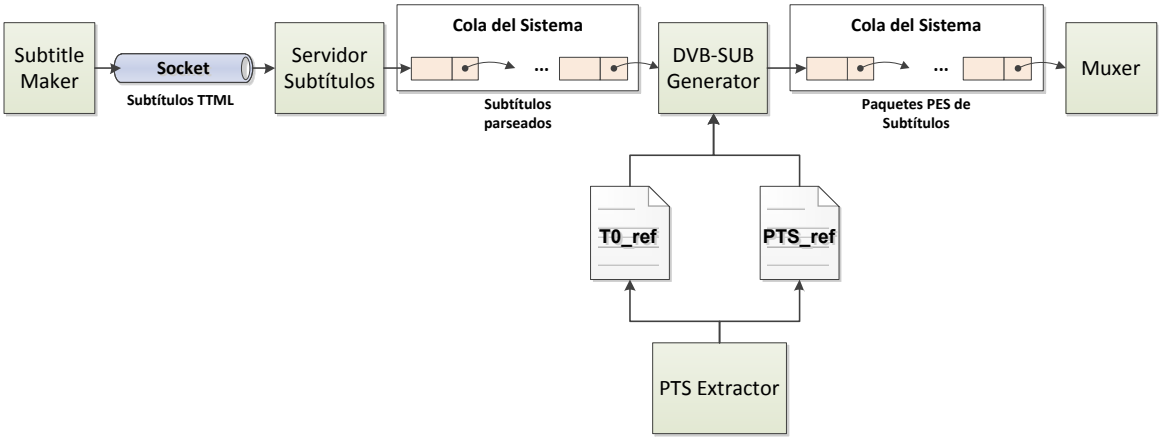


Figura 9.5: Interfaces del DVB-SUB Generator.

Como se puede apreciar, los sistemas de comunicación escogidos son muy diferentes, adaptándose el sistema a la forma de comunicación más conveniente en cada caso. Los apartados subsecuentes describen cada una de las interfaces que se muestran en la figura 9.5.

9.4.1.1. Subtitle Maker - Servidor de Subtítulos

La comunicación entre el Subtitle Maker y el Servidor de Subtítulos se realiza mediante un socket TCP. Como se puede deducir por el nombre, la parte servidora reside en el Servidor de Subtítulos mientras que la parte cliente se aloja en el Subtitle Maker.

Durante la inicialización de ambos módulos, el Servidor crea un socket en una dirección y puerto determinados conocidos por el Subtitle Maker, al que éste último trata de conectarse. Una vez que están conectados, prosiguen su ejecución permaneciendo abierto el socket a lo largo de toda la ejecución del sistema.

Toda vez que el Subtitle Maker obtiene un subtítulo, lo envía en formato TTML modificado a través del socket. Al otro lado del socket, el Servidor de Subtítulos se encuentra esperando a que le lleguen los subtítulos. Cada vez que recibe uno, invoca al Parser para que extraiga la información y genere un Parrafo (el struct descrito en la subsección 9.2.9 en la página 135) para ser enviado al DVB-SUB Generator. Tras enviarlo, vuelve a esperar a que se reciba algo por el socket.

9.4.1.2. Servidor de subtítulos - DVB-SUB Generator

En el caso de la comunicación entre estos dos módulos se optó por utilizar una cola de sistema. De esta forma nos aseguramos de que cada vez que el DVB-SUB Generator extraiga un subtítulo de la cola, éste va a ser el más antiguo de los que ya están preparados.

Por defecto, **en una cola de sistema, pueden coexistir mensajes cuyo tamaño no supere en un momento dado, el valor fijado por el parámetro del kernel, msgmnb, cuyo valor es 16384 bytes**. Puesto que este valor es muy pequeño para el volumen de datos que es producido por el sistema, es necesario cambiar este valor antes de iniciar todo el SSAS. Actualmente este valor está fijado a 1.000.000 de bytes (Apróx. 1 MB) para asegurar el correcto funcionamiento del sistema[4].

Dado que la única tarea del DVB-SUB Generator consiste en la generación continua de paquetes PES de subtítulos, es necesario tener controlados aquellos puntos en los que esta cola se pueda quedar vacía, o de lo contrario, esto puede llevar a fallos de segmentación durante la ejecución. Para tal efecto se ha instalado un manejador de interrupciones en el DVB-SUB Generator que capture la señal de usuario de sistema denominada SIGUSR1 y que ha sido renombrada como SIGNAL_DVBSUB. Así, cada vez que el Servidor de Subtítulos envía un subtítulo, avisa al DVB-SUB Generator lanzando la señal SIGNAL_DVBSUB.

Por su parte, el DVB-SUB Generator comprueba cada vez que va a acceder a la cola, el número de mensajes que hay en la misma. Si hay cero mensajes, se duerme esperando a que salte la interrupción y pueda reanudar su ejecución. De lo contrario, continúa procesando el subtítulo. De esta forma conseguimos no sólo evitar problemas de acceso a la cola, sino también optimizar el uso de la CPU, ya que cada uno de estos módulos son procesos independientes que tratan de acceder al procesador.

Cuando el sistema termina su ejecución, es necesario liberar todos los recursos del sistema, por lo que el DVB-SUB Generator debe eliminar las colas que han sido instanciadas. Puesto que la forma en que el SSAS termina su ejecución es mediante la señal de sistema SIGINT, es decir, cuando el usuario activa el atajo de teclado "Ctrl + c", el DVB-SUB Generator debe instalar otro manejador de interrupciones para dicha señal. Este manejador se encargará únicamente de eliminar la cola entre el Servidor de Subtítulos y el DVB-SUB Generator y finalmente, de terminar la ejecución del mismo.

[4]Este valor está sobredimensionado dado que el sistema actualmente está en periodo de pruebas. Su valor final es susceptible de ser reducido y ajustado teniendo en cuenta el peor caso que se pueda dar en volumen de información, al mismo tiempo, en una cola de sistema.

9.4.1.3. PTS Extractor - DVB-SUB Generator

Esta es la interfaz más sencilla de todas pues se basa en archivos de configuración. Dado que el PTS Extractor únicamente actúa al inicio de la ejecución del SSAS, antes de que el DVB-SUB Generator haya sido lanzado, no hay ningún problema de acceso a variables compartidas, y de ahí que se puedan comunicar mediante este sistema tan sencillo. El PTS Extractor, tal y como se explicó anteriormente, genera dos archivos de configuración donde deposita PTS_{ref} y $T0_{ref}$ respectivamente. Cuando el DVB-SUB Generator necesita estos datos, simplemente tiene que abrir los archivos y leer la información que en ellos se haya contenida.

9.4.1.4. DVB-SUB Generator - Muxer

Al igual que la interfaz entre el Servidor de Subtítulos y el DVB-SUB Generator, esta interfaz está representada por una cola de sistema. Sin embargo, a diferencia del caso anterior, entre ambos módulos no es necesario configurar una señal para coordinar su ejecución. Esto se debe a que el Muxer siempre tiene tareas que ejecutar y no necesita esperar a que haya algo en la cola. Según ha sido diseñado el SSAS, si al acceder el Muxer a la cola, ésta se encuentra vacía, el Muxer lo interpreta como que no hay subtítulos que insertar por el momento, por lo que continúa insertando en el flujo de salida, los paquetes correspondientes a los otros flujos elementales o tablas de información PSI/SI.

En cambio, cuando la cola no está vacía, el Muxer lee el primer mensaje de la cola para poder tener acceso al PTS del paquete PES del subtítulo y poder compararlo con los PTSs de los paquetes PES de vídeo subsecuentes. En el momento en que reciba un paquete de vídeo cuyo PTS sea superior al del subtítulo, el Muxer inserta todos los paquetes de transporte existentes en la cola que pertenezcan al mismo paquete PES, repitiendo este proceso en lo sucesivo. Este hecho es posible, gracias a que cuando el DVB-SUB Generator termina de codificar un subtítulo, encapsula toda la información en paquetes de transporte que se envían a la cola de manera consecutiva hasta que se envía toda la información. Así, cuando el Muxer detecta que hay que insertar el siguiente subtítulo, no tiene que esperar paquete a paquete (paquetes de transporte) pues ya se encuentran todos en la cola. Este proceso de encapsulado será explicado con mayor detalle en el Capítulo 12 en la página 169.

9.5. Conclusiones relativas a la Arquitectura

Inicialmente, una de las tareas más complejas de abordar en el proyecto, era conseguir la perfecta sincronización de los subtítulos a la salida del SSAS, ya que había que guardar la referencia de las transcripciones respecto al audio, a lo largo de toda la cadena de procesos del sistema. Por esta razón, se intentó diseñar un sistema

de subtitulado cuyo principal objetivo fuese facilitar esta tarea y la integración de los diferentes módulos, sin que esto implicase aumentar la complejidad de otras tareas, a priori, más sencillas de llevar a cabo.

Como consecuencia de este diseño, se ha llegado a una solución por la cual la sincronización finalmente es una tarea sencilla, que se resume en calcular los PTS en base a los tiempos de referencia calculados a partir del contenido de los archivos de configuración y los tiempos recibidos del Subtitle Maker, e insertar los paquetes en los puntos adecuados del flujo de transporte con los flujos sincronizados. Estos archivos de configuración son un recurso de implementación que nos permite obtener las equivalencias entre el reloj interno del sistema y el reloj interno de la señal DVB-T sintonizada.

El hecho de que esta tarea sea tan sencilla, es debido a la distribución de la funcionalidad necesaria para llevar a cabo la sincronización, así como al sistema de interfaces entre los bloques desarrollados bajo este proyecto y los módulos que los rodean. A pesar de que dichas interfaces se encuentran entre el límite del ámbito de este proyecto y del SSAS, es necesario incluirlas dentro del proyecto pues es donde han sido desarrolladas, siempre teniendo en cuenta el mejor tipo de comunicación en cada caso y los mecanismos adicionales necesarios para establecer una comunicación controlada de los recursos existentes (instalación de manejadores de señal, envío de señales, etc).

Por todas estas razones, se ha querido dedicar un capítulo entero a explicar todo el entramado de bloques e interfaces para a continuación proseguir con la explicación del resto de trabajos realizados.

Capítulo 10

DVB-SUB Generator. Codificación basada en Píxeles

10.1. Introducción

A lo largo de este capítulo vamos a describir los trabajos realizados durante el desarrollo del software cuyo propósito es generar un flujo de subtítulos a partir de la codificación basada en píxeles. Para iniciar la codificación, es necesario disponer de la imagen que representa al subtítulo. En cambio, lo que este módulo recibe del Servidor de Subtítulos es un Struct con la información en formato texto, por lo que, tal y como ya hemos mencionado, va a ser necesario generar la imagen de cada subtítulo recibido y prepararla para que sea apta para ser tratada por el codificador DVB-SUB.

Es importante mencionar que parte de la funcionalidad de este bloque se basa en el código fuente de dos programas de la librería libavcodec: *bmp.c* y *dvbsub.c*. Sin embargo, dichos programas no son estables ya que no se han actualizado ni mantenido conforme han avanzado las versiones de la librería, por lo que no cumplen con su funcionalidad. Por esta razón fue necesario un estudio detallado de los mismos para subsanar los errores que presentaban y adaptar el código fuente de forma que fuese útil para nuestra labor.

Puesto que los resultados obtenidos durante las primeras fases no fueron del todo convincentes (visualmente), fue necesario añadir una serie de mejoras al código fuente, para que la imagen mostrada por el decodificador TDT fuera de calidad y no presentase errores gráficos.

Todo este proceso de desarrollo es lo que se va a explicar en las secciones siguientes.

10.2. BMP Maker

El BMP Maker es el módulo auxiliar que el DVB-SUB Generator utiliza para generar imágenes a partir del texto de los subtítulos. El código de este módulo está escrito en lenguaje PHP ya que este lenguaje presenta una potente librería para tratamiento gráfico: **php5-gd** [49]. Las características que lo hacen realmente interesante para su uso en este proyecto son las siguientes:

- Soporta una gran variedad de formatos de imagen, entre los que se encuentra el formato BMP.
- Soporta cualquier tipo de **fuente ttf** (*True Type Fonts*).
- Es muy sencillo y la documentación para su uso es extensa, pudiéndose encontrar una gran variedad de ejemplos en Internet.
- Es muy rápido. Prácticamente se podría despreciar el tiempo que conlleva la generación de imágenes mediante este proceso.

Así pues, cada vez que el DVB-SUB Generator requiere convertir el texto en imagen, sólo tiene que hacer una llamada a la función *generaBMP()*, indicando el texto que desea convertir (el texto será la string de caracteres de una de las líneas del Struct Parrafo), el lugar donde desea obtener la imagen y el color del texto del subtítulo. Los colores aceptados por el módulo BMP Maker son todos aquellos especificados en los requisitos del sistema, es decir: azul, verde, cyan, rojo, magenta, amarillo, naranja y blanco, con los que se intenta cubrir los colores habituales en procesos de subtitulado. Además, el código PHP desarrollado contempla la posibilidad de generar las imágenes en base a diferentes letras instaladas en la máquina. Cualquier cambio que se quiera hacer en este aspecto es fácil de realizar. Basta con instalar la fuente deseada e incluirla en el código de la siguiente forma:

```
$tipografia = "/usr/share/fonts/dir_fuente/fuente.ttf
```

Las imágenes generadas por este script son todas del mismo tamaño ($720 \times 44\, píxeles$), sin ningún tipo de compresión y con una profundidad de color de $24\, bits/píxel$. Además, el texto está centrado dentro de la imagen, consiguiendo así imágenes centradas en pantalla y en el tamaño adecuado para cada línea de subtítulos.

Por último, se puede definir el color de fondo de las imágenes. A priori, lo más intuitivo fue fijar este color como blanco. No obstante, tal y como veremos a lo largo de la sección de mejoras introducidas, en el apartado 10.4.2, existen alternativas que producen mejores resultados durante el tratamiento posterior de la imagen.

10.3. Decodificador BMP

Este módulo basa toda su funcionalidad en el método *bmp_decode_frame()* del programa bmp.c. Buena parte de este código no es usado ya que las imágenes generadas por el BMP Maker se encuentran en la forma más sencilla y común de las que se pueden dar en un archivo BMP. En esencia, el proceso de decodificación consiste en leer la información relevante contenida en la cabecera del archivo BMP (ya se explicó la estructura de un archivo BMP en el Capítulo 5) y reordenar el array de píxeles para que pueda ser tratado por el codificador de subtítulos. La información relevante extraída de las cabeceras es la siguiente:

- Tamaño del archivo en bytes.
- Ancho de la imagen. Será igual a 720 píxeles.
- Alto de la imagen. Será igual a 44 píxeles.
- Resolución de color. Será igual a $24\,bits/píxel$.
- Tipo de compresión. Será igual a BI_RGB.

Una vez que el decodificador posee toda esta información, estamos en posición de acceder a la primera posición del array de DWORDs que describe la imagen píxel a píxel. Recordando lo que se explicó en el Capítulo 5, esta descripción se realiza partiendo del píxel situado en la esquina inferior izquierda de la imagen, recorriéndola de izquierda a derecha, y fila a fila, desde la última hasta la primera. **Esta ordenación no es válida a la hora de codificar el subtítulo, ya que el codificador necesita que los píxeles estén ordenados tal y como se van a ver en pantalla, para poder realizar la codificación RLC.** Así pues, la siguiente tarea a realizar por el Decodificador BMP es reordenar los píxeles de forma que queden tal y como se describe en la Figura 10.1 en la página siguiente[1].

Como último paso, se realiza una modificación al array de píxeles que forma el Pixel Data ya ordenado[2]. Hemos visto que el ancho de la imagen es 720 píxeles y que la resolución de color es $24\,bits/píxel$, o lo que es lo mismo, $3\,bytes/píxel$. Esto significa que el tamaño de una línea de la imagen es:

$$Line_{size} = \frac{720\,píxeles}{línea} \times \frac{3\,bytes}{píxel} = 2160\,bytes/línea$$

Dado que 2160 es múltiplo de 4, significa que al final de la línea no va a haber padding, por lo que simplemente tendríamos que ir recorriendo de 3 bytes en 3 bytes el array para acceder a los datos. Recorrer la información de esta forma es un tanto

[1] Aunque hablamos de reordenar píxeles, lo que se reordena son las componentes de color BGR asociadas a cada píxel.

[2] En realidad este paso se realiza de manera simultánea a la reordenación de píxeles.

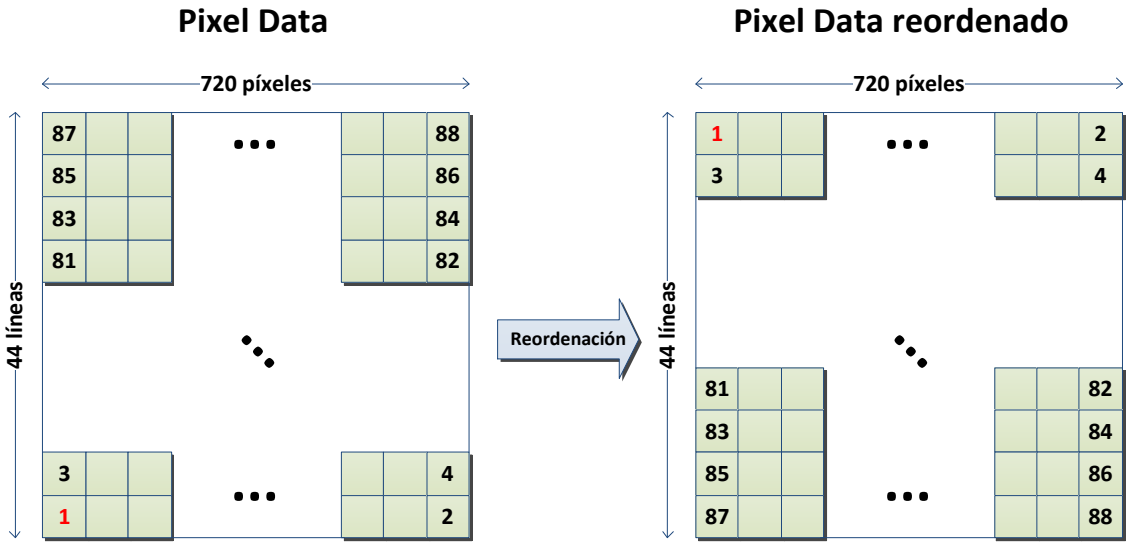


Figura 10.1: Reordenación de los píxeles por el Decodificador BMP.

incómoda, teniendo en cuenta que los tipos de datos existentes en las máquinas son de 8, 16, 32 y 64 bits. Por ello, para que fuese más fácil obtener el contenido RGB de cada píxel y no fuese necesario ir byte a byte, se incluyó un cuarto byte carente de información, cada 3 bytes. De esta forma, un píxel quedaba determinado por 4 bytes en vez de por 3.

Finalmente, el Decodificador BMP deposita el array de píxeles, compuesto en realidad por secuencias de 32 bits (**un array de datos *uint32_t***), en una zona de memoria habilitada para tal efecto y de localización conocida por el codificador de subtítulos.

10.4. Codificación de subtítulos a partir de píxeles

Llegados a este punto, el codificador tiene todo lo necesario para realizar la codificación de un subtítulo: array de píxeles ordenados, tiempo de inicio del subtítulo y tiempo de fin del subtítulo. Con el tiempo de inicio del subtítulo vamos a poder determinar el PTS del paquete PES del subtítulo tal y como se explicó en la sección 9.3 en la página 137. En cambio, el tiempo de fin del subtítulo, en vez de usarlo para determinar el valor de time-out de ese paquete PES, lo vamos a usar para construir una página de borrado, es decir, otro paquete PES compuesto únicamente de un segmento de página y de tantas regiones como tuviera el último paquete PES, pero

en este caso sin ningún objeto[3].

En primer lugar vamos a detallar los aspectos destacados de la creación de los segmentos necesarios para construir tanto los subtítulos, como la página de borrado asociada a cada subtítulo. Este paso basa toda su funcionalidad en el programa dvbsub.c. Acto seguido, describiremos las mejoras desarrolladas para perfeccionar los resultados obtenidos.

10.4.1. Generación de Segmentos

A continuación se van a describir las características principales de los segmentos diseñados para crear los subtítulos. Para cada tipo de segmento, distinguiremos entre las características que debe presentar cuando esté destinado a una página de subtítulos y las que debe presentar si se refiere a una página de borrado. Puesto que la cabecera de los segmentos es común a todos ellos, simplemente es necesario destacar que el valor fijado para el campo page_id es el "0x0001", mientras que el valor del segment_type variará en función del tipo de segmento, tal y como se indica en la Tabla 3.3 en la página 60.

10.4.1.1. Page Composition Segment

Una parte importante a definir en estos segmentos es el estado de la página, ya que de ello depende el tratamiento que tenga que hacer el STB sobre la zona de memoria destinada al renderizado de los subtítulos. Por esta razón, el estado de la página se definirá como *mode change,* cuando sea la primera página de subtitulado del programa, y como *acquisition point*, para el resto de páginas. En el caso de las páginas de borrado, éstas siempre tendrán el mismo estado de página: *normal case*. Tal y como se explicó en el apartado 3.4.3 en la página 55, este estado es válido cuando sólo es necesario incluir las regiones cuyos datos hayan cambiado. Por esta razón, en nuestro caso sólo es válido para la página de borrado, ya que la composición de la página es exactamente igual que la anterior, pero variando el contenido de las regiones, que pasan de tener un bitmap a estar vacías.

En cuanto a la descripción de las regiones contenidas en la página volvemos a tener dos posibilidades:

- Página de borrado: no referencia a ninguna región.
- Página de subtítulos: dependiendo de si el subtítulo tiene una o dos líneas, la página hará referencia a una o dos regiones respectivamente, es decir, **se van a definir tantas regiones como líneas tenga el subtítulo**.
    - Dirección horizontal de las regiones: siempre es la 0, es decir, todas las regiones comienzan en el borde izquierdo de la pantalla.

[3]Esto se explicó en la subsección 3.4.5 en la página 58.

- Dirección vertical de las regiones: la dirección de la línea inferior siempre es la 456, mientras que la de la superior es la 412. En caso de que el subtítulo esté compuesto de una única línea, éste siempre se muestra en la línea inferior.

10.4.1.2. Region Composition Segment

Todas las regiones generadas van a tener siempre las mismas dimensiones y características variando únicamente la lista de objetos que presentan. Mientras que las regiones de páginas de borrado carecen de lista de objetos, las regiones de las páginas de subtítulos van a presentar todas un objeto, que representa el bitmap del subtítulo. La dirección horizontal del objeto dentro de la región va a ser la 0, al igual que la vertical, es decir, el objeto comienza en el pixel superior izquierdo de la región. Esto tiene su razón de ser en que las dimensiones de la región están diseñadas de forma que se ajusten al tamaño de la imagen que va a contener.

El resto de características comunes a todas las regiones son las siguientes:

- Dimensiones de la región: $720 píxeles \times 44 líneas$.
- Region level of compatibility: $4 bits/entrada$.
- Region depth: $4 bits$.
- Color de fondo de la región definido como la pseudo-entrada 0 de la CLUT[4].

Como se puede ver, la resolución de color definida de los píxeles contenidos en las regiones va a ser de 4 bits, por lo que la CLUT mínima necesaria va a ser una de entradas de 4 bits.

10.4.1.3. CLUT definition Segment

Este segmento va a contener las entradas en las que se va a basar el equipo decodificador para reconstruir la imagen del subtítulo. Puesto que la CLUT va a ser de 4 bits, tendremos 16 entradas que serán *full range*, es decir, se van a usar los 8 bits de resolución de cada componente de color (Y,U,V y T). Puesto que se planteó la posibilidad de que cada región contuviera subtítulos de diferentes colores, no se podía definir una CLUT común para todos los subtítulos, sino que ésta debía ser creada de forma dinámica dependiendo del bitmap del subtítulo. Esto nos lleva a que una página de subtítulos va a tener tantas CLUTs como líneas de subtítulos.

Inicialmente se planteó la creación de la CLUT teniendo en cuenta únicamente tres colores: el transparente, el negro y el color de la letra del subtítulo. No obstante, las imágenes generadas por el BMP Maker nunca están compuestas solamente de

[4]En realidad el color de fondo nunca se va a usar ya que el objeto ocupa toda la región, es decir, nunca se va a dar el caso de que en una región existan huecos vacíos a rellenar mediante este color.

esos tres colores, ya que el BMP Maker introduce transiciones de color alrededor del borde de las letras para conseguir un efecto de suavizado en las mismas, tal y como se puede apreciar en la Figura 10.2, en la que se muestra el zoom de un ejemplo de imagen generada por el BMP Maker:

Figura 10.2: Suavizado de las letras en la imagen BMP.

Este suavizado mejora la calidad de la imagen generada por el BMP Maker, pero desacredita el uso de este método, ya que en realidad, la imagen BMP no va a estar compuesta por 3 colores, sino que va a presentar un número superior de ellos. Dado que el número de colores en la mayoría de casos era mayor que 16, es decir, mayor que los colores que pueden ser contemplados con una CLUT de 16 entradas, fue necesario buscar soluciones que nos permitiesen obtener los resultados esperados, sin tener que pasar por hacer uso de CLUTs de 256 entradas, es decir, era necesario buscar soluciones que no implicasen aumentar el ancho de banda necesario. De esto hablaremos en la subsección 10.4.2, donde entre otras cosas, se explicarán los diferentes algoritmos diseñados para crear la mejor CLUT posible en cada caso.

10.4.1.4. Object Data Segment

El Object Data Segment contiene el bitmap del subtítulo codificado. Es el segmento de mayor tamaño de todos y el que implica una mayor complejidad. Dado que el método de codificación que se usa en esta parte es el basado en píxeles, el campo *object_coding_method* debe valer "0x00". Respecto a la creación de los pixel-data_sub-blocks() del campo superior y del campo inferior de la imagen, se partió de la base del algoritmo contenido en el código dvbsub.c.

A partir de los fragmentos útiles de ese código, **se desarrolló una nueva versión de tal forma que el comportamiento fuera el esperado**, de acuerdo al algoritmo explicado en el Capítulo 3, y de forma más concreta en el apartado 3.4.7.7. Buena parte de esa sección se dedicó a detallar el caso basado en 4-bit/píxel_code_strings(), que es el mismo que el desarrollado en el proyecto.

Teniendo en cuenta el proceso explicado en esas secciones, se ha desarrollado un algoritmo que, basándose en los colores introducidos en la CLUT, analiza los píxeles de la imagen, en busca de secuencias de píxeles del mismo color. Todos aquellos píxeles que se encuentren fuera de las letras deberán ser codificados como colores transparentes para que únicamente se vean las letras, cubriéndose la menor cantidad posible de pantalla.

Dependiendo de si se trata de píxeles transparentes, o si son píxeles con color, así como de la cuenta acumulada de píxeles consecutivos del mismo color, el algoritmo seguirá alguna de las vertientes que se muestran en la Figura 3.27 en la página 75 y que se encuentran resumidas en la Tabla E.1 en la página 241 del Anexo E.

El algoritmo de codificación RLC está ligado a los colores de la CLUT asociada a la región que va a contener el objeto. Es importante que la tabla de colores presente en dicha CLUT sea accesible por el algoritmo RLC, para que éste sepa en todo momento, cuál es la entrada del color del píxel correspondiente. Además, la tabla CLUT debe ser coherente con el bitmap que va a hacer uso de ella, ya que de nada sirve conocer la CLUT, si los colores que en ella se hayan contenidos son totalmente diferentes a los de la imagen.

En el apartado 10.4.2 se explicarán de forma cronológica las mejoras introducidas en el proceso de codificación y la forma en que se vieron afectados, tanto el proceso de creación de la CLUT, como el de los objetos.

10.4.1.5. End of Display Segment

Tal y como vimos a lo largo del Capítulo 3 en la página 23, este segmento está vacío, es decir, está compuesto únicamente por la cabecera y señala el fin del paquete PES de subtítulos. Así pues, constituye la última parte del proceso de codificación del subtítulo, justo antes de que el paquete PES sea encapsulado en paquetes de transporte.

10.4.2. Mejoras introducidas

A continuación detallamos el proceso de evolución de la codificación de subtítulos.

10.4.2.1. CLUT Dinámica

Como consecuencia de los malos resultados obtenidos tras la primera versión del codificador (imagen del subtítulo pobre, tamaño elevado del subtítulo, etc.), en una primera instancia se planteó la mejora en este apartado descrita. En vez de basarnos en una CLUT compuesta únicamente de tres colores, la idea era introducir dos colores de forma fija: el color transparente en la entrada 0 y el color negro para el borde de los subtítulos en la entrada 1, y el resto de entradas, asignarlas de forma dinámica en función de los colores presentes en el bitmap del subtítulo.

El algoritmo diseñado se basa en buscar píxel a píxel, colores diferentes a los ya introducidos en la CLUT en cada momento. En cuanto se encuentra un color diferente a los ya contenidos en la CLUT, éste se introduce en la misma, y se actualiza una tabla donde guardamos los colores presentes en la CLUT para facilitar el acceso a los mismos durante el proceso de codificación RLC. Así pues, esta tabla temporal es en realidad un array de 16 posiciones que simula los contenidos de la CLUT, en la que, en vez de tener los colores en formato YUV, éstos se encuentran en formato RGB.

En el momento en que se rellenan las 16 entradas de la CLUT, el algoritmo termina la búsqueda de nuevos colores y continúa con el resto de la codificación. A pesar de que esta mejora nos permite tener una mayor cantidad de colores en la CLUT, seguimos sin cubrir todos los colores presentes en el bitmap, dado que sólo disponemos de 16 entradas.

Una vez iniciada la codificación RLC, el proceso es el siguiente (explicado de forma simplificada):

1. Buscamos cuál es el siguiente píxel a procesar y realizamos la cuenta de píxeles consecutivos.

2. Puesto que los píxeles del bitmap son los decodificados a partir de la imagen BMP, éstos van a estar definidos como colores RGB.

3. Buscamos el color en la tabla temporal. Así, evitamos tener que buscar en la CLUT, ya que implicaría tener que convertir los colores a YUV. Esto no representa ningún problema, ya que los colores contenidos en ambas tablas coinciden posición por posición. En caso de que el color del conjunto de píxeles encontrado no se encuentre en la CLUT, por defecto vamos a asignar la entrada 0, es decir, dejamos esos píxeles como transparentes.

4. Introducimos en el segmento del objeto (en el campo superior o el campo inferior, según corresponda) la sintaxis necesaria para el color introducido y la cuenta calculada, siguiendo el algoritmo descrito en la Figura 3.27 en la página 75.

5. Continuamos con el siguiente píxel (Vuelta al paso 1).

Visualmente, este algoritmo mejoró mucho los resultados obtenidos. Sin embargo, el hecho de asignar por defecto la entrada 0 a los colores no presentes en la CLUT, seguía produciendo imperfecciones en la imagen de los subtítulos, lo que nos llevó a desarrollar la siguiente mejora.

10.4.2.2. Selección de color más cercano

Hasta el momento, durante el algoritmo RLC, cuando un color no se hallaba entre los colores de la CLUT, asignábamos por defecto la entrada 0. Con el fin de mejorar los resultados obtenidos mediante esta técnica, se introdujo una variación en el algoritmo RLC. Esta variación consiste en buscar el color más cercano al del píxel y asignar el color encontrado, en vez de asignar de forma fija la entrada 0, salvo que ésta constituya el color más próximo. El criterio seguido para obtener el color más cercano se basa en el cálculo de la distancia acumulada absoluta entre las componentes RGB de los colores, es decir:

$$distancia = |R_{pixel} - R_{CLUT}| + |G_{pixel} - G_{CLUT}| + |B_{pixel} - B_{CLUT}| \qquad (10.1)$$

La función encargada de determinar el color más cercano, repite el cálculo de la distancia entre el color del píxel cuyo color se desea determinar y todos los colores de la CLUT. Cada vez que la distancia mejora respecto a los casos anteriores, la función almacena ese color como posible candidato. El candidato final será aquel que de lugar a la menor distancia, es decir, el color más cercano al píxel objetivo.

10.4.2.3. Descarte de colores en la CLUT

Continuando con la fase de mejoras, el paso siguiente fue modificar de nuevo el comportamiento del fragmento de código encargado de crear la CLUT. Hasta ahora, el algoritmo había mejorado creando una CLUT dinámica en la que se podían encontrar por un lado, el color transparente y el negro, y por otro, los siguientes primeros 14 colores diferentes del bitmap. No obstante, según la forma de las letras y cómo se produzcan las transiciones entre colores, es muy posible que esos primeros 14 colores acaben siendo muy parecidos, y por consiguiente, redundantes y poco valiosos. Llevado a un caso extremo, si durante la búsqueda de píxeles diferentes, se produce una transición en la que hay 14 tonos muy parecidos del mismo color de forma consecutiva, la CLUT va a presentar muy poca variabilidad, obteniéndose resultados muy parecidos a los que obtendríamos con la CLUT original de 3 colores.

Para solucionar esto, se implantó una solución semejante a la del apartado anterior. Ésta consiste en el descarte de colores para la CLUT que, aunque sean diferentes a los ya introducidos en la CLUT, sean muy parecidos a ellos. De esta forma aumenta el abanico de tonos contenidos en la CLUT y por consiguiente, mejora notablemente

la calidad del subtítulo generado.

Como criterio de descarte se fijaron unos umbrales para cada uno de los tonos. Teniendo en cuenta que los valores RGB van desde 0 hasta 255, se fijaron los siguientes valores:

$$Umbral_R = 20\,;\, Umbral_G = 40\,;\, Umbral_B = 20$$

Como se puede observar, no se han fijado los mismo umbrales para las tres componentes. El umbral fijado para la componente verde es mayor, es decir, se permite una mayor distancia en esta componente, dado que al ojo humano le resulta más difícil distinguir entre tonos verdes parecidos, que entre los tonos rojos o azules.

Según esto, todo color candidato a estar en la CLUT, cuyas componentes se encuentren por debajo de estos umbrales, respecto a alguno de los colores ya presentes en la CLUT, será descartado. Dicho de otra forma, si nos imaginamos un eje de coordenadas 3D simulando las componentes RGB y centrado en el color de la CLUT con el que deseamos hacer la comparación, **cualquier color que se encuentre dentro del prisma de base rectangular cuyas dimensiones en cada eje sean iguales a los umbrales fijados, será descartado**, mientras que los que estén fuera para todos los colores existentes en la tabla de colores, se insertarán en la CLUT. La Figura 10.3 describe la situación arriba explicada.

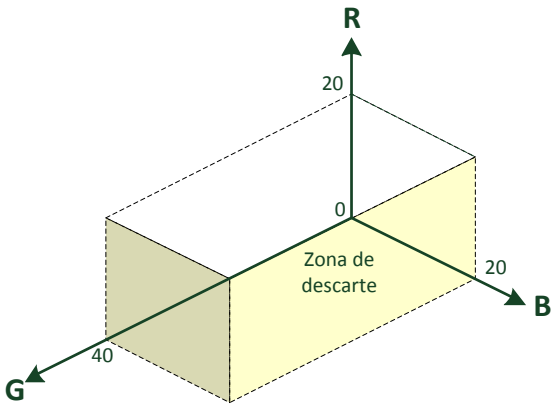


Figura 10.3: Umbrales de descarte de colores en la CLUT.

Por otro lado, la Figura 10.4 refleja el resultado final que se consigue (suponiendo que se diera la secuencia de píxeles planteados en el ejemplo), según apliquemos el procedimiento que hemos llamado como CLUT dinámica, o el de descarte de colores en la CLUT.

Como se puede apreciar, mediante el método de descarte de colores, sólo ocupamos una entrada de la CLUT para una secuencia de colores muy parecidos, mientras

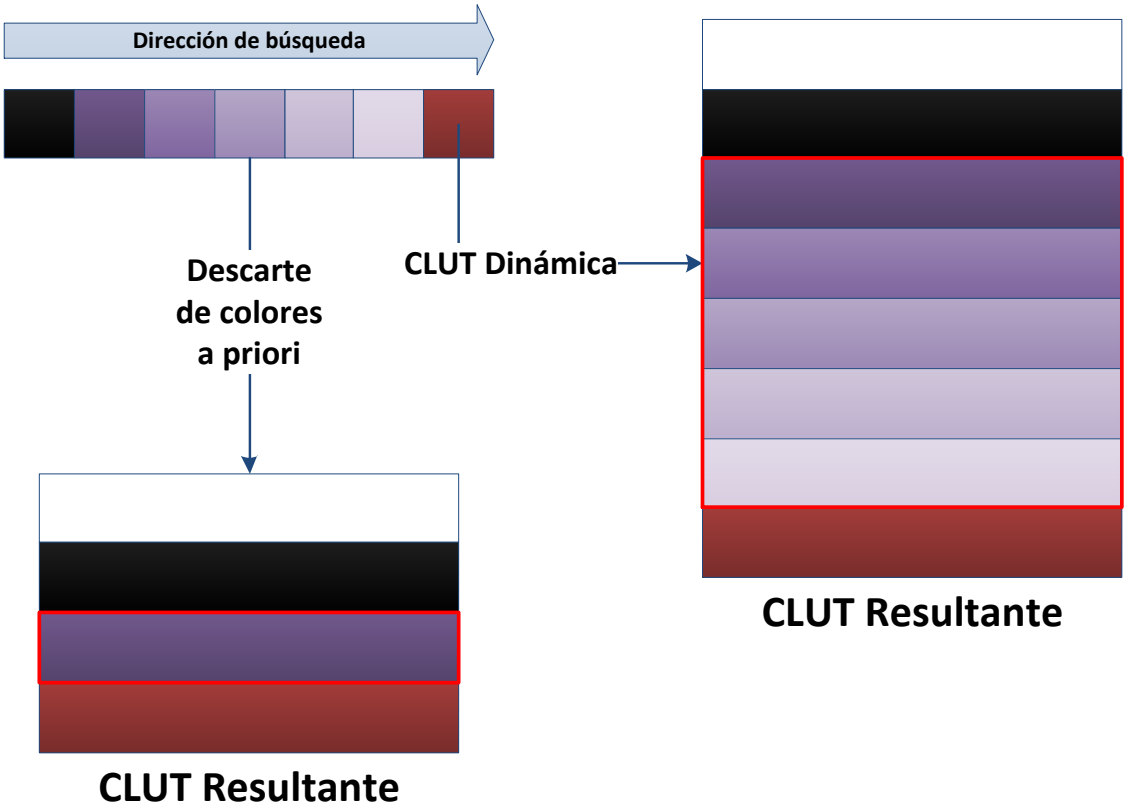


Figura 10.4: CLUT Dinámica vs. Descarte de colores en la CLUT.

que mediante el otro método, esa misma secuencia ocupa 5 entradas. De esta forma se liberan 4 entradas para otros colores y se codificaría esta secuencia de colores muy parecidos, mediante el mismo color. Aunque estemos introduciendo pequeños errores en la imagen, estos apenas son perceptibles por el ojo humano debido al reducido tamaño de un píxel y a las limitaciones del ojo en la detección de variaciones leves de color adyacentes.

En resumen, este procedimiento nos permite obtener una CLUT más rica en tonos, procedimiento que resulta válido gracias a las limitaciones físicas del ojo. Hay que tener en cuenta que el procedimiento explicado en el apartado anterior, se mantiene presente en el código, complementando a esta nueva mejora, es decir, dado que seguimos sin disponer de todos los colores del bitmap en la CLUT, sigue siendo necesario buscar el color más próximo cuando uno de los colores no esté en la CLUT. Sin embargo, ahora la CLUT va a disponer de una mayor variedad de colores por lo que existe una mayor probabilidad de que obtengamos un color que realmente sea parecido al del siguiente grupo de píxeles a codificar.

10.4.2.4. Reducción del payload del ODS

Esta mejora, a diferencia de las anteriores, no tiene como propósito perfeccionar la imagen obtenida, sino reducir el tamaño de los pixel-data sub-blocks. Surge como consecuencia de la mejora que hemos denominado como *selección de color más cercano* en la subsección 10.4.2.2, por la cual, cuando el codificador no encuentra el color de un grupo de píxeles dentro de la CLUT, busca el que más se aproxima.

Según esto, en un momento dado, durante la codificación de un grupo de píxeles es posible que el siguiente grupo, cuyo color es diferente, acabe siendo codificado como el mismo color, si se da la circunstancia de que ese grupo no está en la CLUT y se parece mucho al anterior. Hasta ahora, en las situaciones en las que ocurría esto, a pesar de que ambos grupos resultaban siendo codificados haciendo uso del mismo color, se codificaban como grupos diferentes. Esta mejora busca resolver esta situación, unificando aquellos grupos de píxeles que, a pesar de presentar diferente color, acaban siendo codificados con el mismo. De esta forma, se unifica todo ese conjunto de píxeles bajo la misma cuenta[5], consiguiéndose una reducción notable del payload del segmento.

10.4.2.5. Selección del color de fondo del subtítulo

Para terminar con la sección de mejoras, en este apartado se va a explicar el último método aplicado para corregir aún más los posibles fallos gráficos derivados de la codificación y de la limitación de 16 colores impuesta por la CLUT.

El planteamiento seguido fue realizar diversas codificaciones variando el color de fondo de las imágenes generadas por el módulo BMP Maker. Desde el principio se asumió que la decisión óptima era tomar como color de fondo el color blanco y que, cuando el codificador lo encontrase durante la codificación RLC lo asignase a la entrada 0, es decir, al color transparente. Esto implicaba que no se podría utilizar el color blanco en las letras de los subtítulos puesto que de lo contrario, las letras obtenidas serían completamente transparentes. Así pues, era necesario introducir nuevas modificaciones ya que es inaceptable que no se pueda usar el color blanco en un subtítulo, teniendo en cuenta que es uno de los colores más frecuentes.

En un primer momento, para solucionar esto, se cambió el color de fondo a otro color. Inicialmente el color seleccionado fue un blanco no puro que identificamos como el nuevo color de la entrada 0. Sin embargo, dado que durante la codificación se descartan los colores parecidos a los existentes en la CLUT, los resultados obtenidos siguieron siendo malos.

[5]En realidad, el máximo número de píxeles que se pueden contar mediante un sólo 4-bit/pixel_code_string es 280, que es el caso resumido en la Tabla E.1 en la página 241, como 0000|1|1|11|RRRRRRRR|CCCC, donde RRRRRRRR es el número que sumado a 25 indica el número de píxeles que han de codificarse como el color de la entrada CCCC de la CLUT. En caso de que la cuenta total acumulando varios grupos sea mayor que 280, sería necesario más de un 4-bit/pixel_code_string, es decir, sería necesaria más de una cuenta.

Para conseguir que se pudieran obtener subtítulos en color blanco era necesario generar un color de fondo suficientemente alejado del color blanco para que no se confundiera con este color, es decir, había que asegurar que ese color de fondo se encuentra fuera de los umbrales fijados durante el descarte de colores para la CLUT. La solución fácil sería fijar un color completamente diferente al blanco, como por ejemplo el azul oscuro, pero en ese caso, el color que daría lugar a problemas sería el azul o el negro. Si en cambio se decidiese establecer como color de fondo un tono concreto de rojo, entonces probablemente se obtendrían muchos fallos con las letras de color rojo, y así sucesivamente con todos los colores que han sido contemplados por el BMP Maker.

Finalmente se buscó un color de fondo que estuviese lo más alejado posible en el eje de coordenadas, respecto a los colores contemplados por el BMP Maker. De esta forma, el color de fondo no influiría en el descarte de colores, al no parecerse a ninguno, y haría difícil que durante la búsqueda del color más cercano, para los casos en los que el color no se encuentra en la CLUT, el resultado más cercano fuera dicho color de fondo.

El color de fondo seleccionado fue un tono de color marrón como el mostrado por la Figura 10.5, en la que se puede observar un ejemplo de imagen generada por el BMP Maker.

Figura 10.5: Ejemplo de imagen generada por el BMP Maker.

Esta mejora no sólo habilitó el uso del color blanco como color de letra, sino que aumentó aún más la calidad del subtítulo, al eliminar el ruido que se generaba en torno al borde de las letras. Toda la evolución conseguida mediante este conjunto de mejoras serán puestas de manifiesto en el Capítulo 13, dedicado a describir las pruebas y resultados obtenidos.

10.4.3. Esquema de segmentos según el carácter de la página

Para terminar, resumimos el esquema de segmentos de las páginas creadas a partir del software desarrollado, en función de la finalidad de las mismas. La Figura 10.6 recoge los dos casos posibles: páginas de subtítulos y páginas de borrado.

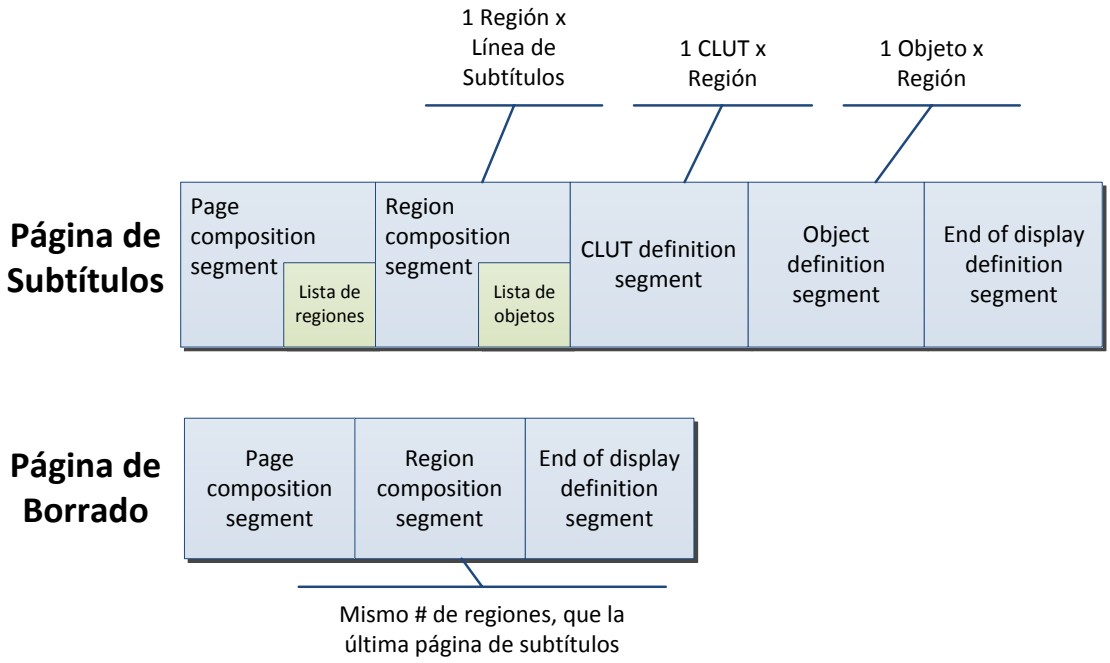


Figura 10.6: Esquema de segmentos según el carácter de la página.

Capítulo 11

DVB-SUB Generator. Codificación basada en cadenas de caracteres

11.1. Introducción

De forma análoga al capítulo anterior, en el actual vamos a describir el proceso seguido durante el desarrollo del módulo encargado de generar un flujo de subtítulos a partir de la codificación basada en cadenas de caracteres.

A diferencia de la codificación basada en píxeles, ésta no necesita generar una imagen como proceso previo a iniciar la codificación, sino que toma directamente la secuencia de caracteres recibida por el Servidor de Subtítulos, además de los tiempos para llevar a cabo su sincronización.

Toda la funcionalidad desarrollada en esta parte fue íntegramente implementada sin contar con el apoyo de librerías auxiliares. Asimismo, es necesario tener en cuenta las restricciones derivadas de este tipo de codificación y que ya hemos mencionado en los requisitos del sistema, en la sección 8.1 en la página 125.

Como última fase del proyecto se incluyó una mejora para que los subtítulos saliesen centrados en pantalla al ser reproducidos los flujos mediante el reproductor VLC.

Todo este proceso de desarrollo es lo que se va a explicar en las secciones siguientes.

11.2. Codificación de subtítulos a partir de cadenas de caracteres

Cada vez que el Servidor de Subtítulos recibe información desde el Subtitle Maker, ésta es mandada al codificador de subtítulos a través de la cola, instante en el que se inicia el proceso de codificación de dicho subtítulo. En este caso, el punto de

partida es el siguiente: cadena de caracteres de las líneas de subtítulos contenidas en el Struct Parrafo, tiempo de inicio del subtítulo y tiempo de fin del subtítulo.

Con el tiempo de inicio del subtítulo vamos a poder determinar el PTS del paquete PES del subtítulo de la misma forma en que explicamos en la codificación basada en píxeles. Sin embargo, el tiempo de fin del subtítulo sí que se ha usado en esta codificación, para calcular el campo time-out del mismo paquete PES y no para el PTS de la página de borrado. La única razón por la que se tomó esta decisión fue por simplificar el sistema, ya que según los requisitos del proyecto, simplemente se pretende demostrar la viabilidad de esta codificación. En cualquier sistema real, debido a las limitaciones de los STB actuales, no tendría sentido utilizar la codificación basada en caracteres, por lo que se centró la atención del proyecto en otros aspectos más importantes, como la integración del sistema aquí desarrollado en el SSAS así como en las tareas necesarias para lograr la sincronización de los subtítulos.

Dicho esto, procedemos a detallar los aspectos destacados de la creación de los segmentos necesarios para construir los subtítulos, a la que seguirá la descripción de la mejora introducida para lograr el centrado de los subtítulos en pantalla.

11.2.1. Generación de Segmentos.

Muchos de los aspectos de la generación de los segmentos son semejantes a los de la codificación basada en píxeles, por lo que haremos un breve resumen en aquellos puntos que ya hayan sido explicados en el capítulo anterior.

11.2.1.1. Page Composition Segment

Durante la generación de este segmento se debe tener en cuenta el número de líneas indicado por el campo num_subs del Struct Parrafo, tal y como se puede observar en la subsección 9.2.9 en la página 135. En función de este número, determinaremos la posición de las regiones y la cantidad de ellas en la página, es decir, una o dos regiones.

11.2.1.2. Region Composition Segment

Este segmento presenta algunas diferencias respecto al de la otra codificación. Como ya indicamos en el capítulo anterior, las regiones sólo van a contener un objeto en el que irá codificado el subtítulo. Este objeto puede ser de diferentes tipos en función de la codificación seleccionada. Así pues, dado que la codificación de este objeto va a ser la basada en caracteres, el campo object_type deberá valer “0x01”, es decir, será un objeto básico de tipo carácter.

Además, es necesario añadir dos campos más a este segmento, que en la codificación basada en píxeles no aparecen. Éstos son el foreground_pixel_code y *el*

background_pixel_code, los cuales hacen referencia a la entrada de la CLUT que va a contener el color de letra y el color de fondo respectivamente[1]. En nuestro caso fijamos sus valores a las entradas 0 y 1 de la CLUT, cuyo contenido se explica a continuación.

11.2.1.3. CLUT definition Segment

A diferencia de la codificación basada en píxeles, la generación de este segmento no entraña una gran complejidad, entre otras cosas porque no es usada por el VLC durante la renderización de las letras, tal y como ya hemos explicado. Así pues, la CLUT introducida es una tabla de colores de 4 bits cuyo valor es siempre fijo, independientemente del subtítulo que se quiera introducir.

Respecto al contenido de la tabla, sus entradas son muy parecidas a las contenidas en la CLUT de TVE, salvo la entrada 1, que se modificó para contener el color deseado de letra (de forma ideal, ya que VLC no lo tiene en cuenta).

11.2.1.4. Object Data Segment

En primer lugar, es necesario indicar que el tipo de codificación del objeto va a estar basada en caracteres. Eso significa que el campo object_coding_method va a valer "0x01". Dado que se trata de la codificación basada en cadenas de caracteres, la estructura de este objeto cambia respecto a la codificación basada en píxeles, tal y como se puede observar en la Figura 3.24 en la página 70. En vez de indicar la información del mapa de bits, ahora lo que se indica es el número de caracteres del subtítulo, y a continuación, los caracteres que componen ese subtítulo codificados en UTF-8, es decir, indicamos el tamaño de la cadena de caracteres y el contenido de la cadena de caracteres.

Atendiendo a la codificación UTF-8 [50], se puede observar que se pueden clasificar los caracteres en dos grupos: los de 1 byte y los de 2 bytes. Además, si observamos el principio de la tabla UTF-8, las primeras 32 entradas están destinadas a caracteres de control (caracteres desde el "0x00" hasta el "0x20"). Por otro lado, tal y como vimos durante la subsección 3.4.7.7 en la página 69, el campo character_code donde se indica el código del carácter, tiene un tamaño de 2 bytes.

Con toda esta información, la forma en que los caracteres son codificados es la siguiente:

1. Se calcula la longitud del subtítulo y se introduce dicho valor en el campo number_of_codes del segmento.

2. Si el siguiente carácter es de un byte, es decir, su valor en hexadecimal se encuentra entre "0x20" y "0x7F", el contenido del campo character_code es

[1]No obstante, esta información es omitida por el VLC, que renderiza los subtítulos siempre con color de letra blanco y fondo transparente.

el siguiente:

a) $character\_code_{1er\,byte} = 0x00$.

b) $character\_code_{2o\,byte} = valor\_UTF_8$.

3. Si el siguiente carácter es de dos bytes, cada uno de ellos se interpreta como un character_code diferente, por lo que es necesario insertar dos character_codes por cada byte. El proceso es el siguiente:

   a) $B1\_character\_code_{1er\,byte} = 0xFF$.

   b) $B1\_character\_code_{2o\,byte} = 1er\_byte\_UTF_8$.

   c) $B2\_character\_code_{1er\,byte} = 0xFF$.

   d) $B2\_character\_code_{2o\,byte} = 2o\_byte\_UTF_8$.

4. Vuelta al punto 2 hasta codificar todos los caracteres del subtítulo.

La mejor forma de entender cómo se codificaría un carácter de dos bytes es mediante un ejemplo. Supongamos que el siguiente carácter a introducir es la "á" cuyo valor UTF-8 es C3 A1. La Figura 11.1 muestra el proceso a seguir.

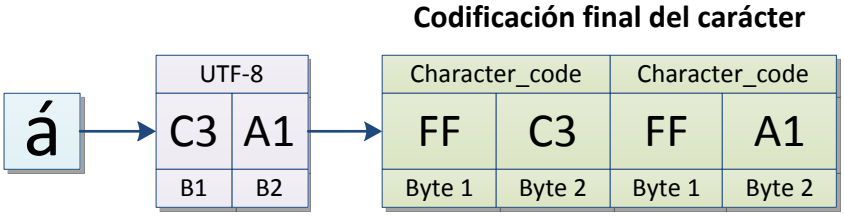


Figura 11.1: Codificación de un carácter de dos bytes a partir del código UTF-8.

De esta forma, el reproductor VLC es capaz de discernir los casos en que el carácter es de un byte de los casos en los que está compuesto por dos bytes. La solución es subóptima ya que se desaprovecha el uso de los 2 bytes de los que ya dispone el campo character_code, pero es la forma en que VLC necesita recibir los caracteres codificados.

Como ya se explicó en el apartado 4.2.2.2 en la página 87, durante la fase de estudio del estado del arte, se logró obtener un ejemplo de DVB-SUB mediante este tipo de codificación y haciendo uso del reproductor VLC. Como ya dijimos, la salida generada fue estudiada mediante un editor hexadecimal a través del cual se analizó el proceso de codificación seguido por VLC. Puesto que el objetivo es conseguir que este reproductor visualice los subtítulos, tuvimos que adaptar nuestra solución para

que contemple esta forma de realizar la codificación, en vez de la contemplada por la norma, según la cual únicamente habría que incluir los bytes dentro de un único character_code.

Tras finalizar el proceso de codificación de los subtítulos, pudimos comprobar que la salida producida no aparecía centrada en pantalla, sino que empezaba justo en el borde de la misma. En la subsección siguiente procedemos a explicar la mejora introducida en el código para resolver esta situación.

11.2.2. Mejoras

Dada la poca flexibilidad permitida por VLC, para este tipo de codificación sólo se planteó la realización de una mejora que asegurase el centrado del subtítulo en pantalla. A continuación se explica dicha mejora.

11.2.2.1. Centrado de los subtítulos

En vistas de que los subtítulos comenzaban al borde de la pantalla, se diseñó un mecanismo que calculase el espaciado necesario a introducir entre el borde de la pantalla y el comienzo del subtítulo. Para ello, en primer lugar era necesario modificar la dirección horizontal del subtítulo, es decir, modificar la dirección horizontal del objeto dentro de la región. Hasta ahora, este campo estaba fijado a 0, y de ahí que todos los subtítulos comenzasen al borde la pantalla.

Sin embargo, para modificar esta dirección, es necesario saber el ancho de todo el subtítulo, al menos de una forma aproximada. La Figura 11.2 refleja el cambio que sufre la posición del objeto, que sería calculada según la Fórmula 11.1.

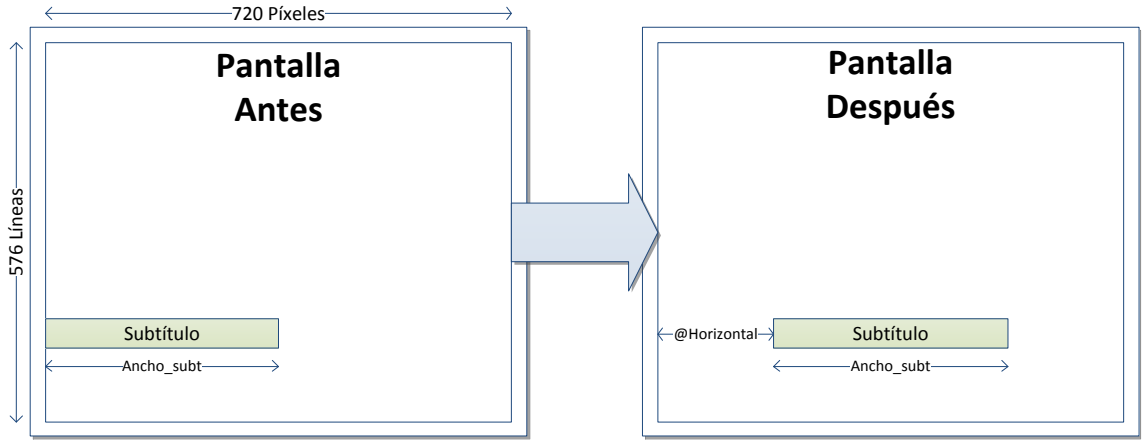


Figura 11.2: Codificación de un carácter de dos bytes a partir del código UTF-8.

$$@Horizontal = \frac{(720 - Ancho_{subt})}{2} \tag{11.1}$$

Donde $Ancho_{subt}$ es el ancho del subtítulo calculado en píxeles. Para obtener este ancho fue necesario realizar una medición del ancho de los caracteres expresado en píxeles, cuyos resultados se muestran en la Tabla 11.1. Dado que un píxel es una medida absoluta, independientemente del tamaño de la pantalla, esta forma de realizar el centrado en pantalla nos permite obtener en todo momento subtítulos centrados, pues el formato de emisión por defecto es 720x576.

Tabla 11.1: Medición del ancho de caracteres expresada en píxeles.

Carácter	Tamaño (px)	Carácter	Tamaño (px)	Carácter	Tamaño (px)
a á à ä â	18.1	A Á À Ä Â	26.1	ç	20.1
b	20.1	B	22.1	Ç	22.1
c	20.1	C	22.1	1	18.1
d	20.1	D	24.1	2	18.1
e é è ë ê	20.1	E É È Ë Ê	20.1	3	18.1
f	14.1	F	16.1	4	20.1
g	20.1	G	26.1	5	18.1
h	16.1	H	22.1	6	18.1
i í ì ï î	6	I Í Ì Ï Î	6	7	18.1
j	6	J	6	8	20.1
k	18.1	K	22.1	9	20.1
l	6	L	18.1	0	20.1
m	30.2	M	26.1	º ª ”	12.1
n	18.1	N	22.1	/ \	14.1
ñ	18.1	Ñ	22.1	¡ ! ’ , ; . : |	6
o ó ò ö ô	20.1	O Ó Ò Ö Ô	26.1	espacio	6
p	20.1	P	18.1	# &	26.1
q	20.1	Q	26.1	$ _ ÷ ~	20.1
r	12.1	R	24.1	%	32.1
s	16.1	S	20.1	( ) [ ] -	10
t	14.1	T	24.1	= + < > ^	24.1
u ú ù ü û	18.1	U Ú Ù Ü Û	22.1	¿ ? { }	16.1
v	20.1	V	24.1	*	18.1
w	26.1	W	34.1	‘	10.1
x	18.1	X	24.1	@	34.2
y ý ÿ	20.1	Y	22.1	·	7
z	18.1	Z	22.1	sp	4

Una vez mapeada esta tabla en el código del programa, calcular el ancho de cada subtítulo es tan sencillo como buscar cada uno de los caracteres que forman el subtítulo e ir sumando el ancho de cada uno de ellos. Para obtener una medida más

precisa, además es necesario sumar tras cada letra, el espaciado existente entre ella y el siguiente carácter, denominado en la tabla como sp. De esta forma se consiguen unos resultados óptimos bajo las condiciones impuestas por el reproductor VLC.

Capítulo 12

Encapsulado en paquetes de transporte

12.1. Introducción

Después de la codificación de los subtítulos, el resultado obtenido en realidad no es un paquete PES, sino que falta por introducir su cabecera, además de los dos primeros campos de datos del paquete PES (data_identifier y subtitle_stream_id).

Una vez generado el paquete PES de subtítulos, el siguiente paso a realizar es la división del mismo en paquetes de transporte, listos para ser incluidos en el flujo de transporte final. Todos los paquetes de transporte correspondientes a un paquete PES se generan en bloque y son entregados al Muxer a través de la cola de sistema correspondiente, donde serán insertados en los puntos adecuados del flujo retardado y sincronizado.

En los apartados siguientes explicaremos la forma en que se generan los paquetes de transporte a partir de la información del paquete PES, así como el proceso por el cual se completa la información del paquete PES que no ha sido producida durante la codificación del subtítulo.

12.2. Paquetes de Transporte

Como ya explicamos en el Capítulo 3, los paquetes de transporte tienen un tamaño de 188 bytes, de los cuales, 4 son de cabecera. Por lo tanto, el proceso de encapsulado va a consistir en dividir el paquete PES en paquetes de 184 bytes a los que habría que añadir la cabecera de 4 bytes.

Dicha cabecera de 4 bytes va a tener algunos campos fijos entre los que hay que destacar el PID, cuyo valor, impuesto por el PMT Maker, va a ser el 7745. La Figura 12.1 muestra algunas de las posibles cabeceras que van a ser generadas durante el proceso de encapsulado en paquetes de transporte.

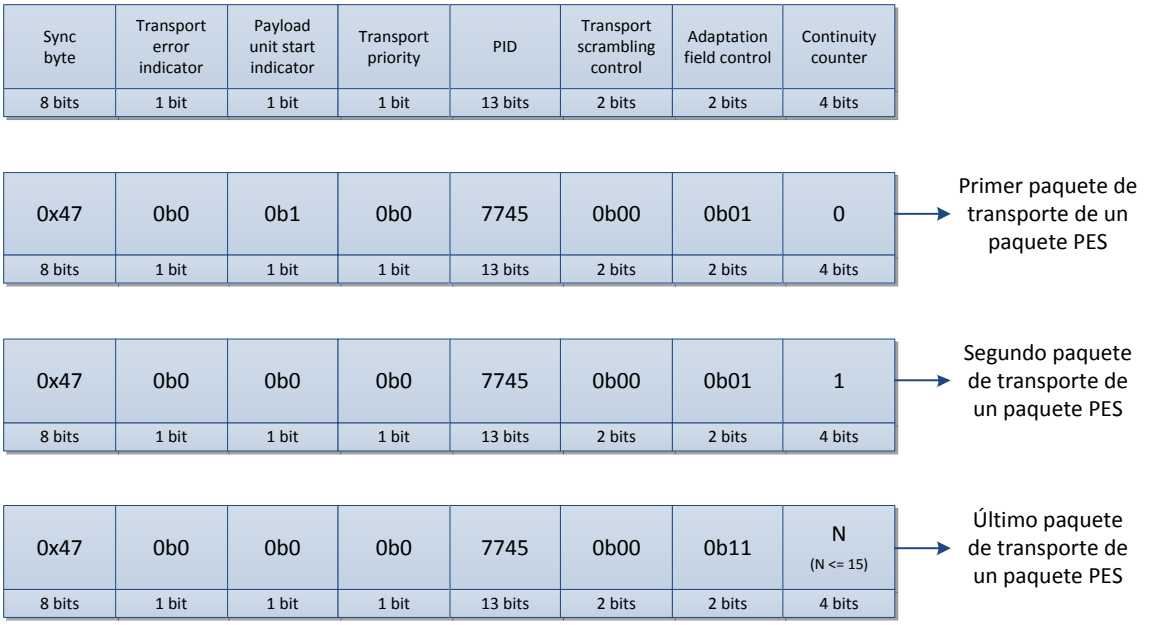


Figura 12.1: Cabecera posibles de un paquete de transporte.

El primer caso de los que se exponen se corresponde con el primer paquete de transporte generado por el sistema. Al ser el primer paquete correspondiente a un paquete PES, el campo payload_unit_start_indicator va a valer 1, y dado que el sistema de subtitulado acaba de iniciarse, el continuity_counter comenzará en 0. Además, suponiendo que se trate de un paquete PES de tamaño mayor que un paquete de transporte, el adaptation_field_control valdrá "0b01", puesto que el primer paquete solo contendrá datos.

El segundo caso refleja la cabecera del siguiente paquete de transporte. En este caso, el campo payload_unit_start_indicator ahora vale 0, ya que el paquete PES no empieza en este paquete de transporte y además, se ha incrementado el valor del continuity_counter. Suponemos que este paquete sólo contiene datos por lo que el valor del adaptation_field_control será el mismo que en el caso anterior.

Por último, el tercer ejemplo se corresponde con el último paquete de transporte de un paquete PES. El campo payload_unit_start_indicator vuelve a valer 0 mientras que el campo continuity_counter se incrementará respecto al último paquete de transporte del paquete PES, presentando siempre un valor entre 0 y 15. Como en la mayoría de los casos, el último paquete de transporte suele presentar bytes de relleno para formar un paquete entero, por lo que el campo adaptation_field_control valdrá "0b11", es decir, presentará datos y stuffing_bytes.

Estos son sólo los casos más típicos, por lo que puede variar la combinación de sus campos. Por ejemplo, en el caso de una página de borrado, que solo ocupa un paquete de transporte, estaríamos ante una cabecera con los siguientes valores:

- $payload\_unit\_start\_indicator = 1$.
- $PID = 7745$.
- $adaptation\_field\_control = 0b11$.

12.2.1. Cálculos importantes

Antes de iniciar el proceso de encapsulado en paquetes de transporte, es necesario conocer una serie de datos imprescindibles para determinar la cantidad de padding a introducir, el número de paquetes a generar, así como poder compppletar algunos de los campos que faltan por determinar de la cabecera del paquete PES. Todos los cálculos necesarios se detallan a continuación.

12.2.1.1. Número de paquetes

El primer paso a realizar durante el proceso de encapsulado es calcular el número de paquetes que se van a generar, ya que este dato va a constituir la condición de parada del algoritmo de encapsulación. Para ello, en primer lugar debemos conocer el tamaño del subtítulo codificado. Este dato es sabido justo al término de la codificación puesto que es el dato devuelto por la función encargada de construir el subtítulo. Hay que tener en cuenta que ese tamaño no se corresponde con el tamaño total del paquete PES, pues el codificador genera únicamente el campo de datos del paquete PES, a excepción de sus dos primeros campos: data_identifier y subtitle_stream_id.

Así pues, para determinar el número de paquetes es necesario conocer el tamaño del paquete PES completo, cuyo valor se obtiene de sumar al valor devuelto por la función de codificación, el tamaño de los campos que faltan por introducir. Dicho tamaño es **16 bytes** y los campos que hay que añadir para completar un paquete PES son: packet_start_code_prefix, stream_id, PES_packet_length, flags, PES_header_data_length, PTS, data_identifier y subtitle_stream_id.

A partir de este dato y del tamaño de la codificación obtenida, el número de paquetes se calcula como sigue:

$$num_{paquetes} = \left\lceil \frac{(Tamaño_{codificación} + 16)}{184} \right\rceil \tag{12.1}$$

12.2.1.2. Número de bytes del último paquete

Se calcula como el resto de la división de la ecuación 12.1. Es necesario para determinar los bytes que faltan por introducir en el último paquete y para determinar la cantidad de bytes de relleno necesarios, que será denominada como padding.

12.2.1.3. Longitud del Padding

Para aquellos paquetes PES cuyo último paquete de transporte no esté completo (esto ocurre cuando $num_{bytes\_ultimo\_paquete} = 0$) , el algoritmo de encapsulación debe introducir bytes de relleno para completar el paquete. Recordando lo visto en la subsección 3.2.7.2, el campo de adaptación destinado a servir de relleno se compone de los siguientes campos: adaptation_field_length, flags y stuffing_bytes. Una vez que sabemos el número de bytes del último paquete, se calcula la longitud del padding, cuyo valor se almacenará en el campo adaptation_field_length, de la siguiente forma:

$$adaptation\_field\_length = 184 - num_{bytes\_ultimo\_paquete} - 1 \tag{12.2}$$

El hecho de restar un byte se debe a que el campo adaptation_field_length indica la longitud del padding expceptuando dicho campo.

12.2.1.4. Longitud del PES

El último cálculo importante a realizar es el que determina el valor del campo PES_packet_length de la cabecera del paquete PES. De igual manera que para calcular el número de paquetes sumamos 16 al tamaño de la codificación, en este caso sólo hay que añadir 10 bytes a dicho tamaño para determinar el valor de este campo. Esto es debido a que el PES_packet_length es la longitud del paquete PES, exceptuando los campos packet_start_code_prefix, stream_id y el propio PES_packet_length.

$$PES\_packet\_length = Tamaño_{codificación} + 10 \tag{12.3}$$

12.3. Paquete PES

Con todos estos estos datos ya calculados, estamos en posición de completar el paquete PES introduciendo su cabecera y los campos data_identifier y subtitle_stream_id del campo de datos. Estos datos se introducirán únicamente al principio del primer paquete de transporte, justo antes de empezar a introducir los datos generados durante la codificación del subtítulo.

Algunos de estos datos son fijos mientras que otros dependen de los cálculos indicados previamente. El valor de estos campos se puede observar en la Figura 12.2 en la página siguiente. De todos los campos que aparecen en la figura, los únicos que no son fijos son el PES_packet_length, cuyo valor es el calculado mediante la ecuación 12.3, y el PTS, cuyo valor se obtiene de la ecuación 9.3 en la página 138.

Tras insertar estos datos, el algoritmo prosigue tal y como se explica en el siguiente apartado.

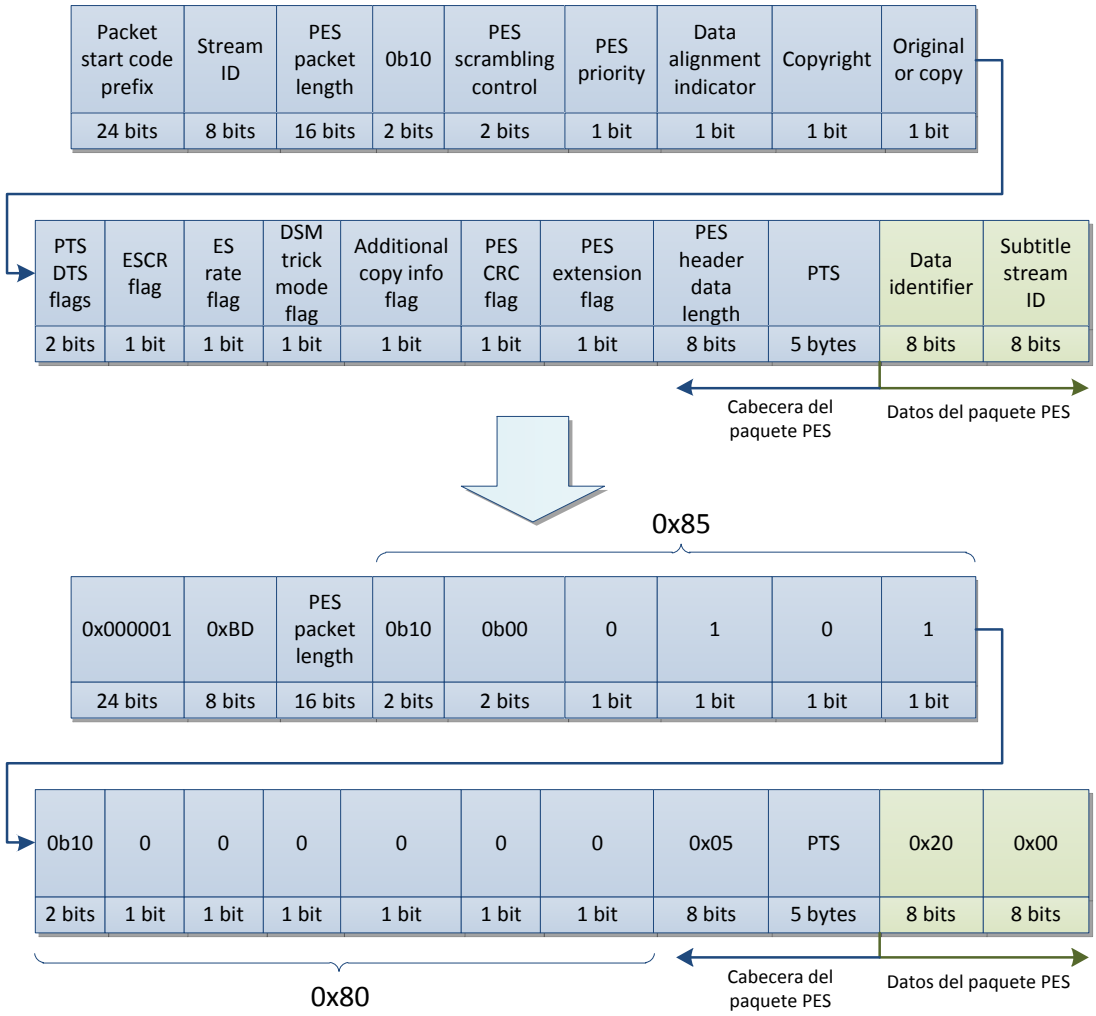


Figura 12.2: Inserción de los campos restantes del paquete PES.

12.4. Diagrama del algoritmo

En los apartados anteriores hemos visto cómo calcular los datos necesarios para que el algoritmo de encapsulación sepa en todo momento el contenido a introducir en los paquetes de transporte, así como la forma en que se debe completar el paquete PES antes de iniciar el encapsulado. En función del número de paquetes, del número de bytes del último paquete, o de si es el primer paquete o el último, el algoritmo toma una serie de decisiones que se resumen en el gráfico de la Figura 12.3 en la página siguiente.

Como se puede observar en la figura, tras construir cada paquete de transporte, el siguiente paso del algoritmo es enviar dicho paquete al Muxer. Por último, resaltar que sólo cuatro de las situaciones posibles suponen el final del algoritmo, mientras que en las otras dos, continúa con la creación del siguiente paquete.

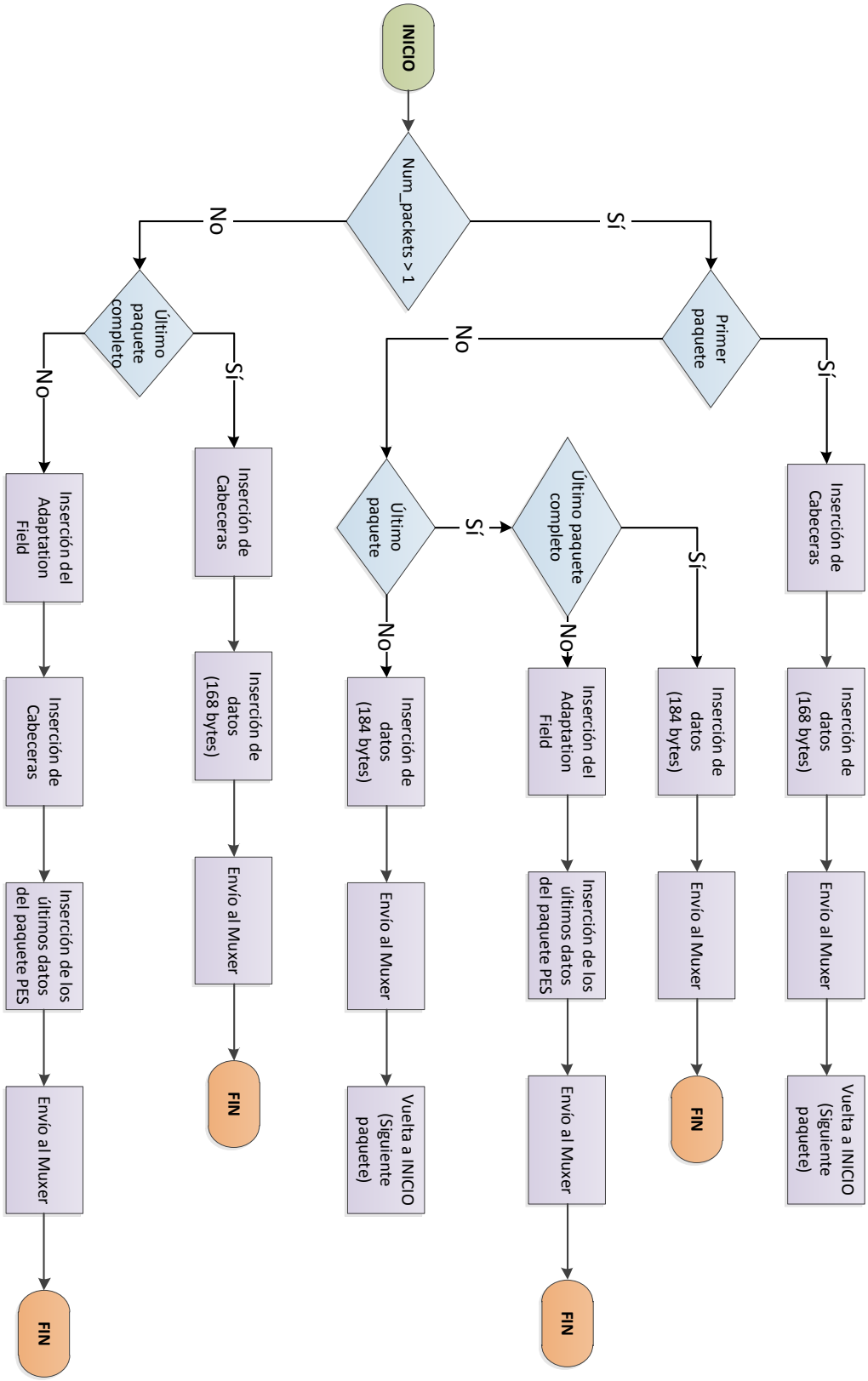


Figura 12.3: Algoritmo de encapsulado en paquetes de transporte.

Capítulo 13

Pruebas y resultados

13.1. Introducción

Una vez que hemos cubierto todas las etapas de desarrollo del proyecto, procedemos a explicar el banco de pruebas propuesto para verificar que se cumplen, no sólo los requisitos del software desarrollado, sino también aquellas características que se han considerado importantes y que hemos querido evaluar para conocer a fondo las prestaciones disponibles. En primer lugar describiremos el entorno de pruebas del que se ha dispuesto a lo largo del proyecto. A esto le va a seguir una descripción de todas las pruebas y resultados obtenidos para ambos tipos de codificaciones y por último, evaluaremos la calidad de los subtítulos obtenidos, así como la sincronización lograda en el SSAS.

13.2. Entorno de pruebas

En esta sección se exponen los elementos de los que se ha hecho uso durante la fase de pruebas del proyecto. Es necesario resaltar la importancia de dicho entorno debido a la dificultad que ha entrañado disponer de un mecanismo fiable que permita evaluar las prestaciones de este proyecto de forma aislada, sin necesidad de incluirlo en el sistema final. De esta forma se consigue acotar las fuentes de errores y se evita depender de la evolución del SSAS.

13.2.1. Software ad hoc

Con el fin de testar el proyecto aquí descrito, se diseñó de forma adicional un software que permitiese visualizar los subtítulos generados sin necesidad de esperar a que el SSAS estuviese desarrollado. Este software se compone de tres módulos que permiten simular el resto del sistema de una forma sencilla. Dichos módulos se describen a continuación.

13.2.1.1. Subtitle Maker Simulator

El Subtitle Maker desarrollado para tal propósito, es una versión muy simple, situada a la entrada del codificador, constituida por un cliente TCP que se conecta con el Servidor de Subtítulos, al que envía la información escrita por la entrada estándar del sistema, es decir, por el teclado. Esto supone que los subtítulos TTML han de ser introducidos a mano y deben estar bien construidos pues en esta parte no se verifica la correcta sintaxis de los mismos. Una vez que el Servidor de Subtítulos recibe un subtítulo TTML entero, éste se parsea y se inicia la codificación del subtítulo.

13.2.1.2. Muxer Simulator

Este módulo representa una versión simplificada del Muxer del SSAS situándose a la salida del codificador. Puesto que no se dispone de los flujos de vídeo, audio y tablas PSI/SI, simplemente se encarga de recibir todos los paquetes del codificador DVB-SUB a través de la cola del sistema, tal y como lo haría el Muxer de la versión final.

A su salida genera un transport stream constituido únicamente por los paquetes del flujo de subtítulos, por lo que al carecer del resto de flujos, no es visualizable en ningún sistema reproductor, ya sea software o hardware. Esta es la razón por la que fue necesario desarrollar el siguiente módulo.

La temporización de este Muxer respecto a la recepción de los subtítulos es algo más estricta que la del Muxer del SSAS, ya que al recibir datos únicamente del codificador de subtítulos, es necesario dormir este proceso mientras en la cola de sistema no haya ningún paquete. Tras la generación de un paquete, el codificador lo manda a la cola del sistema y envía una señal al Muxer para que lo despierte en caso de que estuviera dormido.

13.2.1.3. Insertador de subtítulos

Este programa se complementa con el Muxer construyendo un flujo de transporte con toda la información necesaria para que pueda ser reproducido. Puesto que no disponemos del resto de módulos del sistema, la forma de obtener los flujos de vídeo, audio y tablas PSI/SI es haciendo uso de un flujo de transporte existente guardado en disco y modificando el flujo de subtítulos del mismo. En concreto se trata de un programa que sustituye los subtítulos existentes en un flujo de transporte por los subtítulos creados.

Esta tarea no consiste en la sustitución mera de unos paquetes por otros sino que además se debe tener en cuenta que los paquetes generados deben tener una serie de características:

- Deben tener el mismo PID que el flujo de subtítulos original para que puedan ser identificados por la PMT del flujo de transporte.
- Las marcas temporales de los subtítulos deben tomar como referencia el PTS del primer paquete de vídeo del flujo original.
- El continuity counter debe comenzar por el mismo valor que el primer paquete de transporte del primer paquete PES de subtítulos del fichero original.

Para simplificar la tarea de este programa y de la fase de pruebas, se va a sustituir únicamente el primer paquete PES de subtítulos por uno generado por el codificador. De esta forma el programa deberá tener en cuenta tres situaciones en función de los tamaños de los subtítulos. Si denominamos N al número de paquetes de transporte del paquete PES original y M al número de paquetes de transporte del paquete PES generado, dichas situaciones son las siguientes:

- Si M > N, una vez se hayan intercambiado los primeros N paquetes, se introducen los paquetes restantes del paquete PES generado, de forma consecutiva y justo a continuación del último paquete introducido.
- Si M = N, se sustituyen unos paquetes por otros.
- Si M < N, se sustituyen los primeros M paquetes y a partir de ahí, se eliminan del flujo de transporte nuevo, los paquetes sobrantes del flujo original.

13.2.2. VLC

Es el elemento básico utilizado para visualizar los flujos generados. En el caso de la codificación basada en caracteres es el único medio de que disponemos para verificar que la codificación se ha realizado correctamente dado que es el único visualizador que permite activar los subtítulos codificados como cadenas de caracteres en un flujo ts. En cambio, para la codificación basada en píxeles, disponemos además de la herramienta del apartado siguiente.

13.2.3. LabMU

LabMU es un laboratorio de bajo coste para generación y análisis de señales de Televisión Digital Terrestre en entornos universitarios u otros centros docentes especializados. Aunque los principales beneficiarios de este proyecto serían las Universidades, o cualquier centro de enseñanza especializado, en general esta herramienta es adecuada para cualquier empresa u organismo que tenga como necesidad la implantación de un laboratorio de bajo coste con capacidad multiusuario, escalable y totalmente personalizable [51].

LabMU presenta una interfaz gráfica amigable y fácil de utilizar que facilita, entre otras , las siguientes tareas:

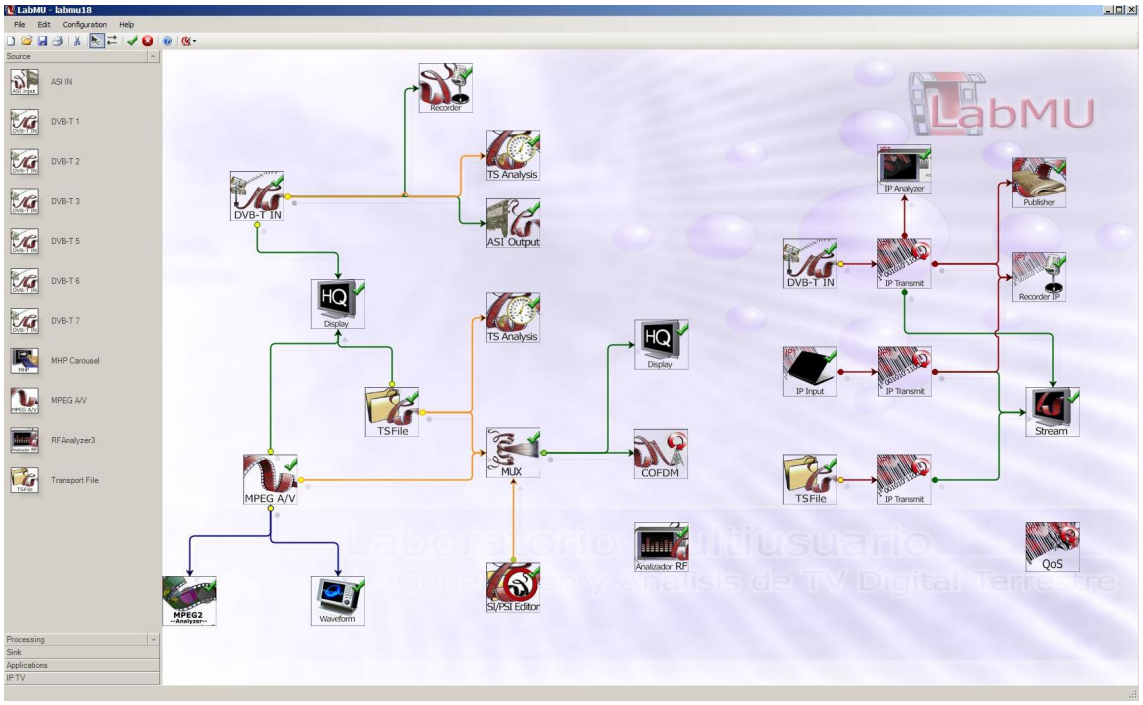


Figura 13.1: Interfaz de LabMU[51].

- Recepción y captura de diferentes fuentes: TDT, IPTV, o cualquier otro estándar de TV basado en DVB.
- Análisis y extracción de componentes de las diferentes fuentes: análisis de tablas SI/PSI, extracción de streams de audio, vídeo o datos, análisis de aplicaciones MHP, etc.
- Streaming de tramas de vídeo en entorno local.
- Composición de nuevas tramas, a partir de una base de datos de componentes, o datos procedentes de otras fuentes.
- Multiplexación y generación de nuevas tramas de transporte personalizadas para ser posteriormente moduladas y transmitidas en entornos locales. Cualquier estándar de modulación de los utilizados en TV Digital será soportado, pero se partirá de la base de los estándares para TDT.

De entre estas características, la que ha resultado especialmente útil ha sido la de poder transmitir los flujos de transporte generados en el entorno local. De esta forma conseguimos emitir dichos flujos y capturarlos mediante un receptor TDT convencional conectado a la antena del laboratorio. Cualquier contenido generado mediante la codificación basada en píxeles debe poder ser reproducido en cualquier receptor por lo que el uso de esta herramienta nos facilitó contrastar los resultados obtenidos.

13.2.4. Equipos STB utilizados

Los receptores TDT utilizados para visualizar la señal generada mediante el LabMU fueron los siguientes:

1. Buy Best Easy Home TDT Black.
2. Energy System T3250.
3. Philips DTR220.
4. Daewoo DSD-800M.
5. MxOnda Mx-STB5289.
6. Lauson 1025N.
7. Gigaset M280 T EPG.
8. Boston DTT 4100.
9. i-Joy i-Vision TDT.
10. AXIL RT160.

Esta selección de 10 reproductores se extrajo de uno de los estudios realizados por el Centro Español de Subtitulado y Audiodescripción [5] con el fin de analizar los mejores receptores TDT en cuanto a accesibilidad, por lo que representan una buena forma de consolidar la validez de los resultados obtenidos.

13.3. Codificación de subtítulos basada en píxeles. Pruebas y resultados

En este apartado vamos a describir secuencialmente las pruebas realizadas y los resultados obtenidos a lo largo del desarrollo de la codificación basada en píxeles. Las pruebas consisten en generar un flujo de subtítulos y, a partir de él y del flujo de transporte guardado en disco, construir un nuevo flujo de transporte mediante el Muxer y el Insertador de subtítulos. Toda prueba descrita fue evaluada mediante los dos sistemas descritos: el VLC y el LabMU.

13.3.1. Resultados iniciales

Dado que los resultados que se obtuvieron inicialmente fueron muy pobres, no se consideraron ni siquiera válidos como codificación por lo que no se profundizó en las pruebas. Debido a la CLUT tan limitada que se construía, los subtítulos generados, en su mayor parte eran transparentes, a excepción de los bordes y de alguna parte coloreada.

13.3.2. CLUT dinámica

En la Figura 13.2 se pueden observar los resultados obtenidos tras aplicar la primera de las mejoras desarrolladas.

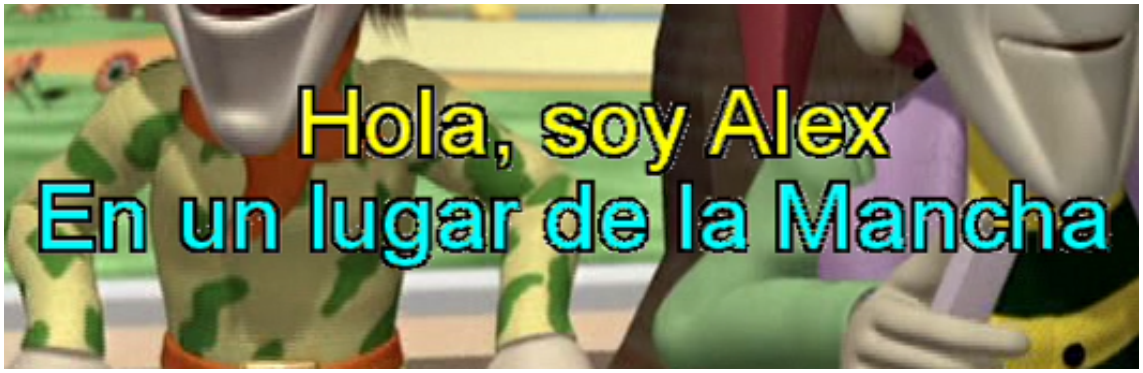


Figura 13.2: Cod. basada en píxeles. Resultados con CLUT dinámica.

Como se puede apreciar, las imperfecciones en el subtítulo son numerosas, pero al menos los subtítulos son reconocibles. Entre dichas imperfecciones, las más importantes son las siguientes:

- Píxeles blancos en zonas donde no debería haber nada, sobre todo en el borde exterior de las letras.
- Contorno negro de las letras irregular.
- Zonas mal coloreadas en el interior de las letras.

Por otro lado, el tamaño del paquete PES de subtítulos obtenido fue de 9.090 bytes, demasiado grande en comparación con los resultados que se van a mostrar a continuación.

13.3.3. Selección de color más cercano

Tras la siguiente evolución implementada en el código, los resultados mejoraron notablemente. Como se puede observar en la Figura 13.3, el contorno de las letras se encuentra mucho más definido que en el caso anterior, es decir, el borde negro se muestra como un trazo continuo, en vez de como trazos con saltos. Respecto al problema de los píxeles blancos, éste se corrige en cierta medida, aunque siguen apareciendo. Por último, el color interior de las letras se ve algo más nítido y uniforme, lo cual cobra sentido, teniendo en cuenta que ahora no asigna la entrada 0 por defecto a los colores que no estén presentes en la CLUT, sino que asigna el más parecido de los que se encuentran en la tabla (esta razón es aplicable también al hecho de que el borde esté mejor definido).

Figura 13.3: Cod. basada en píxeles. Resultados con Selección de color más cercano.

Respecto al tamaño del subtítulo obtenido mediante este método fue de 8.418 bytes. Esto supone un ahorro de 672 bytes respecto al caso anterior, además de la notable mejora gráfica.

13.3.4. Descarte de colores en la CLUT

Tras conseguir resolver el problema de los bordes de las letras quedaba por conseguir eliminar las manchas que aún aparecían en la imagen, así como obtener colores de letras más uniformes. Con la mejora del descarte de colores en la CLUT obtuvimos un gran avance en ambos sentidos tal y como se puede apreciar en la Figura 13.4.

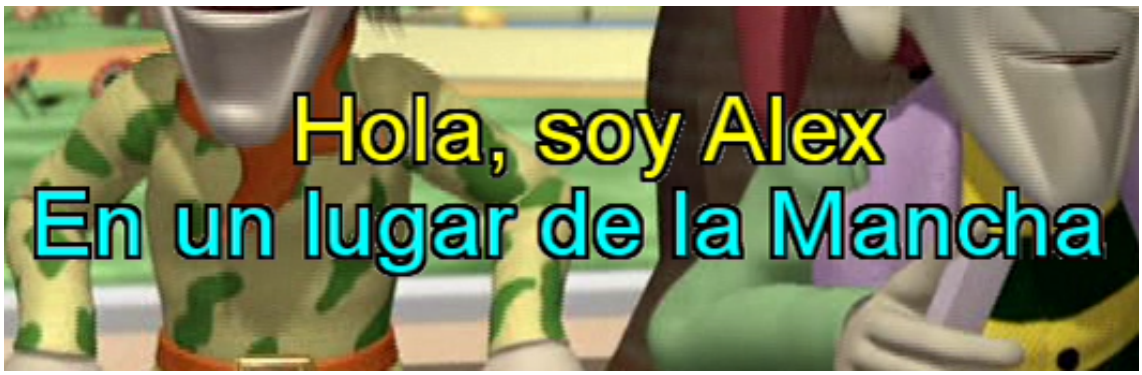


Figura 13.4: Cod. basada en píxeles. Resultados con Descarte de colores en la CLUT.

A pesar de que aún quedan manchas blancas en torno al borde de las letras (en especial en el borde derecho de la letra “y”) los resultados que se lograron con el interior de las letras fueron notablemente mejores que en los casos anteriores. Esto se debe a que la CLUT que se genera ahora contiene colores más representativos que los que contenía hasta ahora. En este caso, el tamaño del subtítulo aumentó ligeramente a 8.506 bytes, es decir, se incrementó en 88 bytes. Aunque esto pueda parecer un inconveniente, en realidad no lo es porque como consecuencia de esta

mejora se pudo introducir otra con el fin de reducir el tamaño del subtítulo. Dicha mejora se explica a continuación.

13.3.5. Reducción del payload del ODS

Al introducir esta mejora en el algoritmo lo único que se buscaba era reducir el payload del segmento del objeto, por lo que era de esperar que no se obtuvieran mejoras gráficas. No obstante incluimos la Figura 13.5, como muestra de que por el contrario, tampoco se produce un empeoramiento de los resultados en términos gráficos.

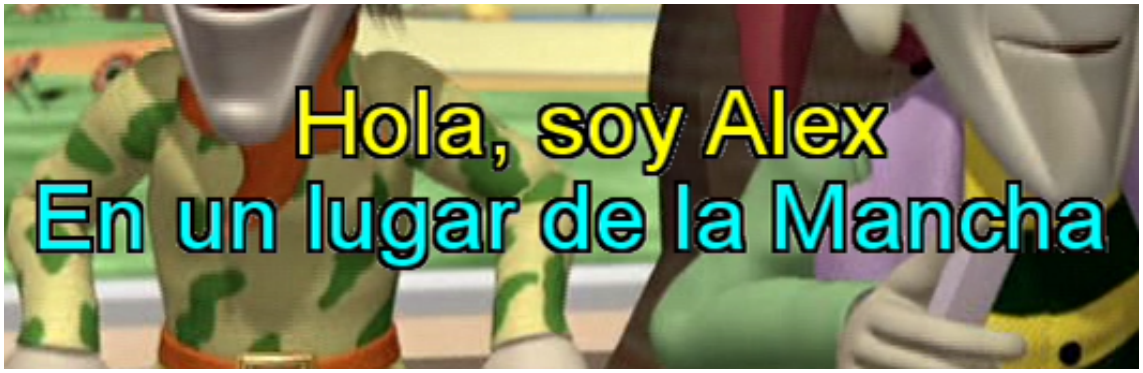


Figura 13.5: Cod. basada en píxeles. Resultados con Reducción del payload del ODS.

Mediante esta nueva mejora el subtítulo obtenido fue de 8.328 bytes, reduciendo el tamaño del mismo respecto al caso anterior en 178 bytes.

13.3.6. Selección del color de fondo del subtítulo

Como última mejora en esta codificación se introdujo aquella por la cual cambiamos el color de fondo del subtítulo generado mediante el BMP Maker. Como se puede observar en la Figura 13.6, el problema de los píxeles blancos desaparece por completo, obteniéndose un subtítulo limpio y en condiciones óptimas para ser mostrado en pantalla.

Por otro lado, el hecho de introducir un color de fondo que fuese más fácil de distinguir por los algoritmos de generación de la CLUT y de búsqueda de colores parecidos, supuso una nueva mejora en términos de tamaño de subtítulo obtenido. El proceso de codificación se optimizó nuevamente en este sentido generándose un subtítulo de 7.993 bytes, lo que supuso una reducción de 335 bytes respecto al caso anterior.

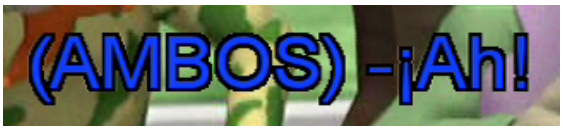
Si comparamos esta última imagen con la obtenida si ningún tipo de mejora, queda de manifiesto que las mejoras introducidas han contribuido a mejorar la calidad de la imagen tanto gráfica como numéricamente.

Figura 13.6: Cod. basada en píxeles. Resultados con Selección del color de fondo del subtítulo.

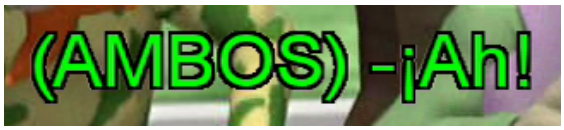
Así pues, **queda de manifiesto la notable mejoría en la calidad de la imagen obtenida, que además se obtiene haciendo uso de 1.097 bytes menos**.

13.3.7. Subtítulos de diferentes colores

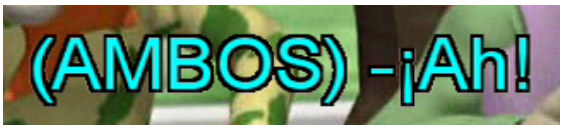
Una vez que obtuvimos subtítulos sin errores gráficos, probamos el comportamiento del algoritmo de codificación ante los diferentes colores de letra contemplados en el BMP Maker, tal y como se puede ver en la Figura 13.7.



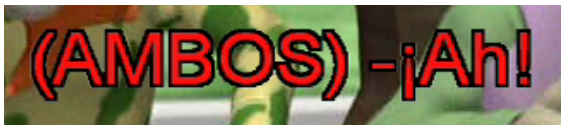
(a) Subtítulo Azul



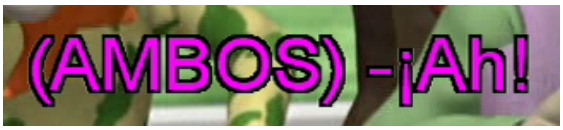
(b) Subtítulo Verde



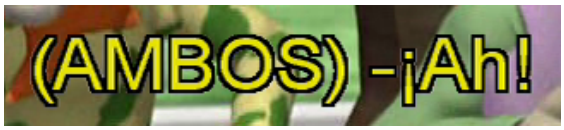
(c) Subtítulo Cyan



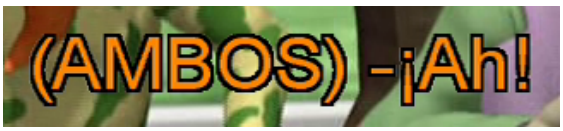
(d) Subtítulo Rojo



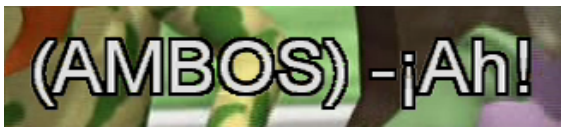
(e) Subtítulo Magenta



(f) Subtítulo Amarillo



(g) Subtítulo Naranja



(h) Subtítulo Blanco

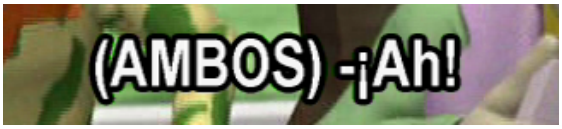
Figura 13.7: Cod. basada en píxeles. Subtítulos en los diferentes colores posibles.

Esta prueba era necesaria para asegurar el correcto funcionamiento de dicho algoritmo ante cualquiera de estos colores, y de hecho, supuso un reajuste de los umbrales fijados para el algoritmo de descarte de colores, hasta llegar a los indicados en la subsección 10.4.2.3 en la página 154.

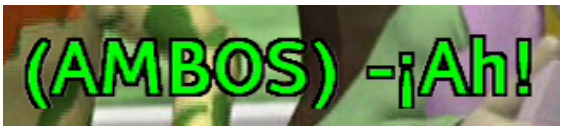
Como se puede observar en las ocho imágenes, para cada uno de los colores se obtienen buenos resultados, lo que nos permite tener un amplio abanico de colores para generar los subtítulos en el color deseado.

13.3.8. Subtítulos con diferentes tipografías

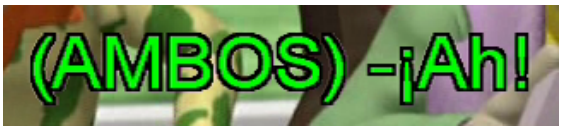
Además de habilitar hasta ocho colores diferentes para generar los subtítulos, se inició una búsqueda con el fin de determinar la tipografía más favorable para generar el subtítulo. Para ello, nada mejor que fijarse en las cadenas de Televisión, y en concreto, en los subtítulos DVB-SUB de TVE. Con el fin de buscar la letra más parecida, se incluye el siguiente grupo de figuras bajo el nombre de Figura 13.8, donde podemos comparar todas respecto a la letra de TVE.



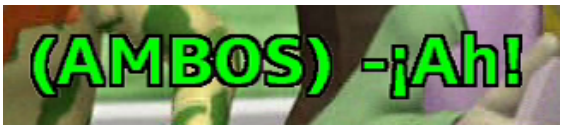
(a) Subtítulo de TVE



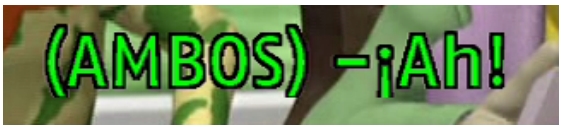
(b) Subtítulo con letra Tiresias



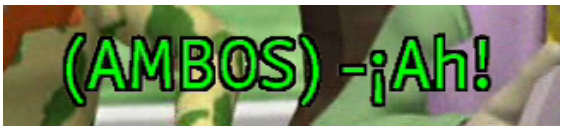
(c) Subtítulo con letra Arial



(d) Subtítulo con letra Mayberry

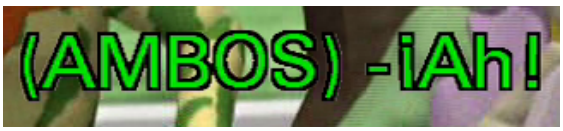


(e) Subtítulo con letra Fago

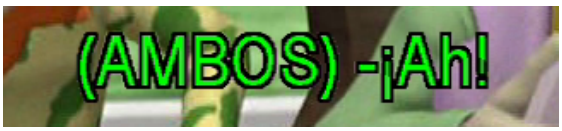


(f) Subtítulo con letra Officina Sans

(g) Subtítulo con letra Vera



(h) Subtítulo con letra Zurich



(i) Subtítulo con letra Arial Narrow

Figura 13.8: Cod. basada en píxeles. Subtítulos en las diferentes letras verificadas.

Por un lado, fijamos nuestra atención en aquellos tipos de letras con las que obtuviéramos un ancho de subtítulo parecido al de TVE. De esta forma, filtramos todas las candidatas a: Fago, Officina Sans y Arial Narrow. Finalmente, de entre estas tres, **seleccionamos Arial Narrow** por ser la más parecida en el trazo de la letra. No es exactamente igual que la de TVE, pero, aparte de que esto no era un requisito, simplemente se trataba de buscar una letra bajo los mismos criterios que los aplicados por las cadenas de televisión, de forma que resulte agradable la lectura de los subtítulos.

En cualquier caso, esta elección no afecta al sistema de ninguna forma, por lo que, en cualquier momento, el tipo de letra se puede sustituir por la que se desee, siempre que se encuentre en formato TTF.

13.3.9. Subtítulos largos

Para corroborar que la elección realizada era adecuada para subtitulado, hicimos una prueba adicional consistente en generar, mediante cada una de las letras expuestas, un subtítulo con la máxima longitud permitida de acuerdo a lo establecido por la norma UNE 153010 [52], es decir, 37 caracteres. Tras definir el patrón del subtítulo de prueba ("It is only a question of nomenclature"), obtuvimos los resultados de la Figura 13.9.

(a) Sub. largo con letra Tiresias

(b) Sub. largo con letra Arial

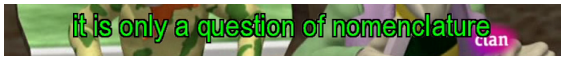
(c) Sub. largo con letra Mayberry letra

(d) Sub. largo con letra Fago

(e) Sub. largo con letra Offina Sans

(f) Sub. largo con letra Vera

(g) Sub. largo con letra Zurich



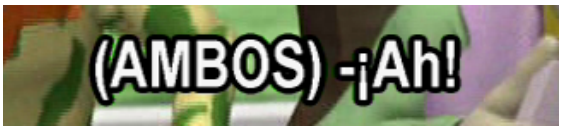
(h) Sub. largo con letra Arial Narrow

Figura 13.9: Cod. basada en píxeles. Subtítulos largos.

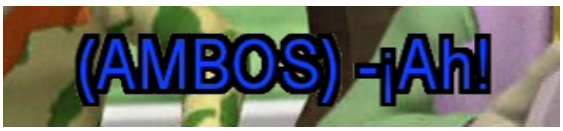
Como se puede apreciar, el tipo de letra Vera no sería válido ya que no permite insertar un subtítulo largo en pantalla, mientras que el resto de tipografías sí lo permiten. Esta prueba sirve además para ratificar la elección realizada en el apartado anterior, ya que la tipografía Arial Narrow, además de ser la más parecida a la de TVE, permite sin lugar a dudas albergar subtítulos largos en pantalla.

13.3.10. Subtítulos Finales

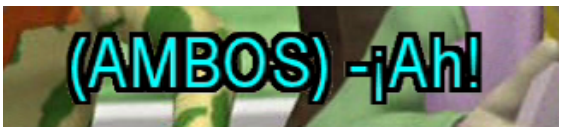
Como consecuencia de todas las pruebas realizadas, se pudo comprobar que los subtítulos generados presentan un borde de menor grosor que la tipografía utilizada por TVE, independientemente del tipo de letra seleccionado. Para hacer aún más parecidos los subtítulos generados a los de TVE, se aumentó dicho grosor obteniendo los siguientes resultados, con los que damos por terminada la fase de pruebas correspondiente a la codificación basada en píxeles:



(a) Subtítulo de TVE

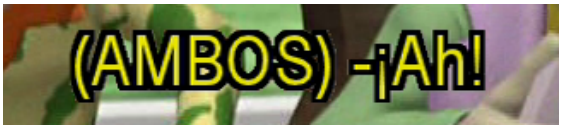


(b) Subtítulo Azul

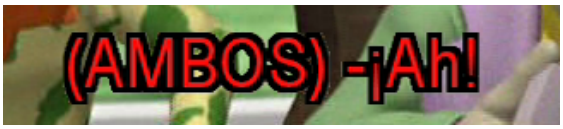


(c) Subtítulo Cyan

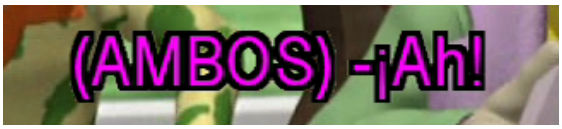
(d) Subtítulo Verde



(e) Subtítulo Amarillo

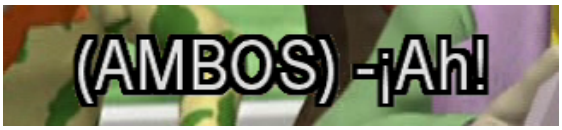


(f) Subtítulo Rojo



(g) Subtítulo Magenta

(h) Subtítulo Naranja



(i) Subtítulo Blanco

Figura 13.10: Cod. basada en píxeles. Subtítulos en los diferentes colores posibles y con letra Arial Narrow (versión Final).

13.4. Codificación de subtítulos basada en cadenas de caracteres. Pruebas y resultados

El banco de pruebas diseñado para evaluar la codificación basada en cadenas de caracteres es mucho más sencillo puesto que las características a valorar son mucho

menores que en la otra codificación. Al igual que en el caso anterior, las pruebas consisten en generar un nuevo flujo de transporte con los subtítulos incluidos, aunque esta vez, el método de visualización va a consistir únicamente en hacer uso del VLC[1]. En los apartados subsiguientes únicamente nos vamos a centrar en los resultados visuales y no en el tamaño de los paquetes PES de subtítulos, ya que no tiene sentido hablar de ello por dos razones:

1. A diferencia de la codificación basada en píxeles, el tamaño del paquete únicamente depende del contenido del subtítulo y no de la codificación que el código haga.

2. Como consecuencia del punto 1, todos los paquetes PES tienen un tamaño de entre 2 o 3 paquetes de transporte, dependiendo de la longitud de la cadena de caracteres.

13.4.1. Resultados iniciales

Como ya se explicó en el Capítulo 11 en la página 161, inicialmente los subtítulos generados se mostraban en pantalla descentrados, tal y como se refleja en la Figura 13.11, donde se muestra, a modo de ejemplo, un subtítulo constituido por las letras del abecedario.

Figura 13.11: Cod. basada en cadenas de caracteres. EL abecedario.

Como ya se explicó, esta situación se subsanó introduciendo en el código una mejora para lograr el centrado de los subtítulos en pantalla, la cual se basaba en estimar el ancho de cada subtítulo en función del contenido del mismo. El siguiente apartado recoge los resultados logrados mediante dicha mejora.

13.4.2. Resultados. Centrado de los subtítulos

En primer lugar, realizamos la misma prueba que en el apartado anterior, pero esta vez, tras incluir la sección de código encargada de realizar el centrado del subtítulo. Por un lado, la Sub-Figura 13.12a en la página siguiente, muestra el abecedario centrado, mientras que la Sub-Figura 13.12b en la página siguiente, presenta un subtítulo real, también centrado.

[1]En realidad, también se realizaron pruebas en receptores TDT convencionales, pero tal y como se indicó en los requisitos, debido a que este tipo de codificación no está contemplada en el software de los mismos, fue imposible visualizarlos.

(a) Cod. basada en cadenas de caracteres. El abecedario centrado.

(b) Cod. basada en cadenas de caracteres. Subtítulo Centrado.

Figura 13.12: Cod. basada en cadenas de caracteres. Subtítulos centrados.

De esta forma se demuestra que el centrado de subtítulos funciona a la perfección, independientemente del contenido del mismo.

13.4.3. Prueba. Ancho máximo permitido

Por último realizamos una prueba para verificar el máximo ancho permitido por esta codificación y su posterior visualización mediante el VLC. Puesto que el ancho no se puede fijar a un número de caracteres, dado que cada carácter tiene un ancho en particular, tomamos como referencia el ancho expresado en dos caracteres diferentes: el carácter "." y el carácter "í", cuyos resultados se muestran en la Figura 13.13.

(a) Cod. basada en cadenas de caracteres. Máximo número de "." consecutivos.

(b) Cod. basada en cadenas de caracteres. Máximo número de "í" consecutivos.

Figura 13.13: Cod. basada en cadenas de caracteres. Prueba de ancho máximo.

En la Sub-Figura 13.13a, se puede observar cómo se pueden incluir hasta un máximo de 63 caracteres ".", mientras que en la Sub-Figura 13.13b, llegan a aparecer hasta 69 caracteres "í".

Esta medida constituye una mera guía para constatar que cualquier subtítulo que sobrepase estos límites no podrá ser visualizado correctamente. Sin embargo, llegados a este punto de la ejecución del programa completo (tanto en codificación de píxeles como de caracteres), se da por supuesto que los subtítulos recibidos desde el Subtitle Maker deben estar bien formateados y adaptados al ancho máximo de pantalla, es decir, 720 píxeles.[2]

13.5. Sincronización de los subtítulos

Para finalizar el capítulo de pruebas, se ha querido hacer una mención a uno de los principales aspectos que este proyecto trata de cubrir: la sincronización. Hasta ahora no se ha incluido ningún apartado en el que demostremos que la sincronización entre los subtítulos, el audio y el vídeo se realiza adecuadamente. Esto se debe a que la sincronización, a pesar de haber supuesto un fuerte control temporal en cada una de las etapas del SSAS, no requiere un control estricto de los tiempos, ya que al ojo humano, le resulta muy difícil distinguir secuencias de imágenes consecutivas separadas temporalmente por unos pocos milisegundos. La precisión de la sincronización está limitada por la precisión de los tiempos que ofrece el reconocedor de voz, por lo que en comparación con el error de los tiempos del reconocedor, cualquier delay en mililsegundos, consecuencia del procesado de nuestros algoritmos, es prácticamente despreciable.

A pesar de este hecho, el sistema desarrollado en su conjunto, tras las pruebas pertinentes en el laboratorio, asegura la sincronización de los subtítulos bajo situaciones controladas. Por situaciones controladas entendemos aquellas que están adecuadas al reconocedor de voz (presencia de un único interlocutor, ausencia de ruido/música de fondo, pronunciación adecuada y pausada, etc.), fuera de las cuales, los tiempos de los fragmentos de texto transcritos no siempre son fiables.

Así pues, en las condiciones actuales, la sincronización entre audio/vídeo y subtítulos depende principalmente de la precisión en la medida de tiempos del software de reconocimiento de habla. Teniendo en cuenta la arquitectura del sistema, y el hecho de que se ha tomado el tiempo obtenido por el ASR como medida para la generación de los PTS, se considera alcanzado el objetivo de sincronización del proyecto en lo que respecta a los módulos desarrollados.

Por otro lado, **una de las características más ventajosas de este sistema es que, al estar modulado, resulta muy fácil extraer el módulo reconocedor de voz por otro mejor, según vaya evolucionando la tecnología**. Esto no afectaría a la precisión de la sincronización lograda por el resto del sistema, sino a la precisión de la transcripción, ya que el sistema únicamente toma los tiempos de referencia que el reconocedor proporciona, para calcular los puntos óptimos de

[2]Este dato cambia en los canales de alta definición.

sincronización mediante el sistema descrito en la sección 9.3 en la página 137.

Por último, mencionar que la codificación de los flujos de subtítulos es prácticamente instantánea (menos de 1 segundo) por lo que se cumple el requisito definido en la sección 8.1 como UR-008.

Capítulo 14

Publicaciones

A continuación se proporcionan los detalles de la publicación relacionada con el proyecto recogido en este documento, así como una breve descripción de las características y objetivos del congreso al que se ha enviado dicha contribución.

14.1. XXI Jornadas TELECOM I+D

Las Jornadas de Telecom I+D [54], que este año 2011 alcanzan su 21ª edición, son un foro de referencia anual para el sector de las Telecomunicaciones y las Tecnologías de la Información. En esta edición se hacen eco del creciente interés que el concepto de Ciudad Inteligente está despertando en la sociedad y de los innumerables retos y oportunidades que este paradigma presenta en el campo de las Telecomunicaciones y las Tecnologías de la Información y el Conocimiento. Bajo el lema "Las TIC en las ciudades del siglo XXI: la SmartCity", este congreso pretende desarrollar unas jornadas que permitan aglutinar tanto los más recientes avances en las distintas tecnologías que habilitan el concepto de ciudad inteligente como disponer de los lugares de encuentro necesarios para la discusión estratégica y el intercambio de experiencias en lo que se refiere a las ciudades del mañana y su papel como ecosistema de innovación.

Con el objetivo de contribuir a dichas jornadas mediante el envío de un trabajo que resuma las principales características y funcionalidad del sistema descrito en este proyecto, se desarrolló un artículo denominado *Subtitulado automático sincronizado DVB* [53] que, tras el proceso de evaluación y revisión correspondiente, ha sido aceptado para su presentación en la sesión de posters de las XXI Jornadas TELECOM I+D.

Capítulo 15

Plan de Proyecto

15.1. Introducción

En este capítulo vamos a describir el plan de proyecto a seguir, comenzando en primer lugar por la descripción de las fases por las que pasará el mismo. A continuación, se explicará de forma resumida el plan de seguimiento a realizar a lo largo del proyecto, así como la forma en que se realizará la gestión del cambio, del riesgo y de la calidad. Acto seguido, se describirá el estándar de codificación utilizado y para terminar, incluiremos la planificación y el presupuesto del proyecto.

15.2. Fases del proyecto

En esta sección se van a describir las etapas de las que consta el proyecto así como los objetivos y las tareas a desarrollar en cada una de ellas. La etapa que ha supuesto un mayor reto en este proyecto ha sido la de implementación, en concreto la fase de desarrollo del sistema codificador de subtítulos basado en píxeles.

15.2.1. Estudio de Viabilidad del Sistema (EVS)

Esta etapa tiene como objetivo el análisis del conjunto concreto de necesidades para proponer una solución a corto plazo, que tenga en cuenta restricciones económicas, técnicas, legales y operativas. Asimismo, se identifican los requisitos que se ha de satisfacer y se estudia la situación actual.

A partir del estado inicial, la situación actual y los requisitos planteados, se estudian las alternativas de solución. Dichas alternativas pueden incluir soluciones que impliquen desarrollos a medida, soluciones basadas en la adquisición de productos software del mercado o soluciones mixtas. Se describe cada una de las alternativas, indicando los requisitos que cubre.

Una vez descritas cada una de las alternativas planteadas, se valora su impacto en la organización, la inversión a realizar en cada caso y los riesgos asociados. Esta

información se analiza con el fin de evaluar las distintas alternativas y seleccionar la más adecuada.

Las actividades que engloba este proceso se recogen en la Figura 15.1.

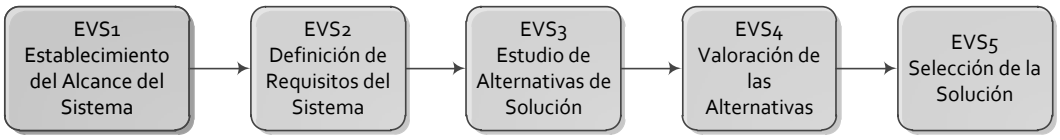


Figura 15.1: Fases del estudio de viabilidad del sistema.

15.2.1.1. EVS1: Establecimiento del alcance del Proyecto

Se estudia el alcance de la necesidad planteada por el cliente o usuario, realizando una descripción general de la misma. Se determinan los objetivos, se inicia el estudio de los requisitos y se identifican las unidades organizativas afectadas estableciendo su estructura. Esta etapa se compone de tres tareas:

1. **Estudio de la solicitud:** se realiza una descripción general de la necesidad planteada por el usuario, y se estudian las posibles restricciones de carácter económico, técnico, operativo y legal que puedan afectar al sistema. Antes de iniciar el estudio de los requisitos del sistema se establecen los objetivos generales del Estudio de Viabilidad, teniendo en cuenta las restricciones identificadas anteriormente.

2. **Identificación del alcance del sistema:** se analiza el alcance de la necesidad planteada. Una vez establecido el alcance, se identifican las unidades organizativas afectadas por el sistema, así como su estructura y responsables de las mismas.

3. **Especificación del alcance del EVS:** en función del alcance del sistema y los objetivos del Estudio de Viabilidad del Sistema, se determinan las actividades y tareas a realizar. En particular, hay que decidir si se realiza o no el estudio de la situación actual y, en el caso de considerarlo necesario, con qué objetivo. En nuestro caso, no se realizó ningún estudio de la situación actual del sistema ya que no existe dicho sistema.

Una vez definido el alcance, éste se formalizará con el cliente, marcando la pauta para la toma de decisiones futuras y realización de actividades a nivel operativo.

15.2.1.2. EVS2: Definición de requisitos del sistema

A lo largo de esta fase se deben determinar los requisitos generales y sus prioridades. Estos se añadirán al catálogo de requisitos sirviendo para el estudio y valoración de las distintas alternativas de solución que se propongan. Se compone de las siguientes tres tareas:

1. **Identificación de las directrices técnicas y de gestión:** se recoge información sobre los estándares y procedimientos relativos a las siguientes áreas:

   a) Gestión de proyectos: seguimiento, revisión y aprobación final.

   b) Arquitectura de sistemas: centralizada o distribuida.

   c) Directrices de Planificación.

   d) Directrices de Gestión de Cambios.

   e) Directrices de Gestión de Calidad.

2. **Identificación de requisitos:** esta etapa es fundamental para la obtención de las necesidades que ha de cubrir el sistema en estudio, por lo que requiere de la participación del usuario.

3. **Catalogación de requisitos:** se analiza la información obtenida en la etapa anterior, definiendo y catalogando los requisitos (funcionales y no funcionales) que debe satisfacer el sistema, indicando sus prioridades. La sección 8.1 en la página 125 contiene un resumen del contenido del catálogo de requisitos.

15.2.1.3. EVS3: Estudio de alternativas de solución

Este estudio se centra en proponer diversas alternativas que respondan satisfactoriamente a los requisitos planteados. Teniendo en cuenta el ámbito (ver sección 1.4 en la página 6) y la funcionalidad (ver sección 8.2 en la página 127) que debe cubrir el sistema, puede ser conveniente realizar, previamente a la definición de cada alternativa, una descomposición del sistema en subsistemas.

En la descripción de las distintas alternativas de solución propuestas, se debe especificar si alguna de ellas está basada, total o parcialmente, en un producto existente en el mercado. El estudio de alternativas de solución se divide en dos tareas:

1. **Pre-selección de alternativas de solución:** una vez definidos los requisitos a cubrir por el sistema, se estudiarán las diferentes opciones que hay para configurar la solución. Entre ellas, se considerará la adquisición de productos software estándar del mercado, desarrollos a medida o soluciones mixtas. Dependiendo del alcance del sistema, puede ser conveniente realizar inicialmente

una descomposición del sistema en subsistemas. Se establecerán las posibles alternativas sobre las que se va a centrar el estudio de la solución, pudiendo incluso combinar las opciones que se consideren más adecuadas.

2. **Descripción de las Alternativas de Solución:** para cada alternativa propuesta, se identificarán los subsistemas que cubre y los requisitos a los que se da respuesta, así como la estrategia de implantación a seguir.

15.2.1.4. EVS4: Valoración de las alternativas

Una vez descritas las alternativas, se realizará una valoración de las mismas, considerando el impacto en la organización y los posibles beneficios que se esperan, contrastados con los costes asociados. Se realiza también un análisis de los riesgos, decidiendo cómo enfocar el plan de acción para minimizar los mismos (posteriormente vamos a realizar su descripción, en la sección 15.5 en la página 201) y cuantificando los recursos y plazos precisos para planificar cada alternativa. A continuación se explican las tareas que comprende, y que han de realizarse para cada una de las alternativas seleccionadas:

1. **Estudio de la inversión:** se valora el impacto en la organización y se establece su viabilidad económica.

2. **Estudio de los riesgos:** se identifican y valoran los riesgos asociados y se determinan las medidas a tomar para minimizarlos.

3. **Planificación de alternativas:** se determina el enfoque más adecuado para llevar a buen fin la solución propuesta en cada alternativa.

Es necesario resaltar que esta fase tuvo especial relevancia en el proyecto debido a la falta de alternativas, estándar o mixtas, existentes en el mercado. Debido a esto, la única alternativa a considerar fue la de llevar a cabo un desarrollo a medida, lo cual se trató de evitar puesto que era la opción que mayores riesgos implicaba.

15.2.1.5. EVS5: Selección de la solución

Esta etapa consiste en una reunión con la Dirección del proyecto durante la que se presentan las distintas alternativas de solución, resultantes de la actividad anterior. En dicha presentación, se debaten las ventajas de cada una de ellas, incorporando las modificaciones que se consideren oportunas, con el fin de seleccionar la más adecuada. Finalmente, se aprobaría la solución o se determinaría su inviabilidad. Esta etapa se compone de tres tareas:

1. **Convocatoria de presentación:** se efectúa la convocatoria de la presentación de las distintas alternativas propuestas y se espera confirmación por parte del Comité de Dirección de las alternativas a presentar.

2. **Evaluación de alternativas y selección:** una vez recibida la confirmación de qué alternativas van a ser presentadas para su valoración, se efectúa su presentación al Comité de Dirección, debatiendo sobre las ventajas e inconvenientes de cada una de ellas y realizando las modificaciones que sugiera dicho Comité, hasta la selección de la solución final.

3. **Aprobación de la solución:** el Comité de Dirección da su aprobación formal o determina la inviabilidad del sistema, ya sea por motivos económicos, de funcionalidad como resultado del incumplimiento de los requisitos identificados en plazos razonables o de cobertura de los mismos.

15.2.2. Planificación

Tras aprobarse la viabilidad de la solución propuesta, se inicia la etapa de planificación, en la cual se pretende abordar los costes temporales y económicos de la ejecución del proyecto. Al final de este capítulo detallaremos el contenido de la planificación propuesta para este proyecto.

15.2.3. Estado del arte

Una vez establecida la planificación del proyecto, y antes de comenzar con la etapa de análisis del sistema, realizaremos un estudio detallado del estado del arte, que ayude al equipo de trabajo a minimizar los esfuerzos y encontrar las herramientas óptimas para llevar a cabo la solución seleccionada durante el EVS. Las tareas que se contemplarán en esta fase son las siguientes:

1. Investigación de las características de los sistemas de subtitulado en tiempo real.

2. Estudio al detalle de la normativa de referencia en subtitulado DVB.

3. Búsqueda de herramientas software para subtitulado DVB.

4. Estudio del formato de imagen BMP.

15.2.4. Análisis

La etapa de análisis se centra en qué se tiene que hacer, es decir, en los requerimientos del sistema desde el punto de vista del usuario, tratando de comprender el problema que se quiere abordar. Durante esta etapa se realizarán las siguientes tareas:

- Definición del modelo de negocio del proyecto, así como el dominio en el que se inscribe.

- Descripción completa de los subsistemas funcionales en los que se divide el conjunto del proyecto.
- De acuerdo a los subsistemas definidos, se establecerá la funcionalidad de cada uno de ellos.
- Especificación de las interfaces existentes entre los subsistemas, ya sean internos al proyecto, o externos, es decir, también se tendrán en cuenta las interfaces que han de ser definidas para que el proyecto desarrollado pueda ser incluido en el sistema total.
- Estudio de la forma más adecuada para realizar la sincronización de los subtítulos.
- Determinación del plan de pruebas a seguir para la evaluación de las unidades funcionales especificadas.

15.2.5. Diseño

La etapa de diseño se va a descomponer en dos fases complementarias: el diseño lógico y el diseño físico, que se realizarán de manera secuencial en el orden en que aparecen.

Ambas fases tienen el propósito de encontrar una solución que cumpla los requerimientos del usuario. Sin embargo, mientras que el modelo lógico del sistema se centra en qué funciones lógicas deben ser implementadas sin tener en cuenta ningún tipo de tecnología, el modelo físico describe qué tecnología se va a utilizar para implementar la solución propuesta en el modelo lógico, además de la manera como se va a aplicar. Los aspectos que se van a cubrir en ambos modelos son los siguientes:

- Diseño de la arquitectura del sistema teniendo en cuenta las unidades funcionales definidas en la etapa de análisis.
- Definición del modelo de procesos a desarrollar para llevar a cabo toda la funcionalidad descrita.
- Definición de los tipos de comunicación a utilizar en las interfaces que han sido definidas durante el análisis del sistema: sistema de colas, interrupciones, sockets, etc.
- Definición del modelo de datos, es decir, descripción de las estructuras de datos que viajarán por las interfaces descritas.
- Diseño detallado de las pruebas a realizar en el plan de pruebas especificado durante la etapa de análisis.

15.2.6. Implementación

La etapa de implementación aborda la codificación de las unidades funcionales descritas anteriormente. La implementación se ha subdividido en las siguientes etapas:

- Codificación del servidor de subtítulos encargado de recibir la información del Subtitle Maker.
- Implementación del Parser para la extracción de la información contenida en los subtítulos TTML.
- Desarrollo del sistema codificador de subtítulos basado en píxeles.
    - Codificación de los módulos de procesamiento previo a la codificación: BMP Maker y Decodificador BMP.
    - Implementación del codificador DVB-SUB basado en píxeles.
- Desarrollo del sistema codificador de subtítulos basado en cadenas de caracteres.
- Implementación del sistema de encapsulado en paquetes de transporte, de la información contenida en los paquetes PES de subtítulos generados.
- Codificación de la parte del Muxer relativa a la comunicación con el generador de subtítulos en formato DVB-SUB.

15.2.7. Pruebas

La fase de pruebas consiste en la evaluación de las funcionalidades desarrolladas durante la fase de implementación. Se realizarán los diferentes tipos de pruebas:

- Pruebas individuales de cada uno de los módulos para evaluar su comportamiento de manera aislada. En algunos casos no será posible ya que algunos de ellos necesitan de las salidas de otros para poder realizar sus funciones, por lo que la evaluación de los módulos de los que dependen deberá ser rigurosa para reducir la probabilidad de introducir errores en fases posteriores.
- Pruebas de conjunto para verificar el correcto funcionamiento tras la integración de todos los módulos. Éstas se podrán llevar a cabo de forma secuencial, tras la integración de un nuevo módulo a los ya desarrollados.
- Pruebas de rendimiento con el fin de poner de manifiesto las capacidades del sistema desarrollado.

- Pruebas de integración en el sistema completo en el que se integrará el proyecto desarrollado.

Con el fin de llevar a cabo todas estas pruebas, en los casos en que sea necesario se desarrollará software adicional para evaluar las prestaciones del sistema. La previsión de SW que será necesario desarrollar es la siguiente:

- Codificación de un Subtitle Maker simplificado que actúe como la parte cliente del socket enviando subtítulos TTML.
- Desarrollo del entorno necesario para probar los subtítulos generados.

15.2.8. Redacción del Proyecto

Esta etapa comprende el periodo de tiempo empleado en redactar toda la documentación en la que se recogen los aspectos importantes del proyecto, tales como el estudio del estado del arte o la descripción de los trabajos realizados.

15.2.9. Implantación

La fase de implantación es la última de todas y se llevará a cabo únicamente, en el momento en que el sistema desarrollado pase todas las pruebas diseñadas de forma satisfactoria. Esta fase se corresponde con la puesta en marcha del sistema de subtitulado en la plataforma del usuario, momento en que también deberá estar completado el resto de módulos del sistema completo.

15.3. Plan de seguimiento

Conforme el proyecto avanza, es importante establecer cómo vamos a monitorizar el progreso del mismo. Para ello, definiremos el marco temporal en el que las actividades deben estar terminadas, para lo cual se definirán una serie de hitos que nos permitirán llevar el control de la evolución del proyecto, subdividiendo la tarea global en segmentos más manejables. Dichos hitos se incluyen en la sección 15.8 en la página 203.

Para la realización del seguimiento se establecen tres granularidades diferentes:

1. Revisión de hitos: cuando se concluyan los hitos se realizará una revisión formal en la que se evaluarán los hitos establecidos en el plan de proyectos en las fechas indicadas en el mismo.

2. Revisión mensual: se realizará, en reuniones presenciales, la revisión mensual del estado del proyecto. Durante estas reuniones se analizará la ejecución de las actividades y el cumplimiento del cronograma comprometido. Asimismo,

se estudiarán los problemas surgidos y los puntos críticos que pueden resultar de interés durante el desarrollo de los trabajos. En caso de existir una reunión por revisión de hitos cercana, se unificarán ambas reuniones.

3. Seguimiento informal: cuando fuera preciso se realizarán cuantas reuniones técnicas sean requeridas.

15.4. Gestión del cambio

La Gestión del cambio de un Proyecto es el proceso en el que se aprueban o rechazan las peticiones de cambios, las acciones correctoras y las acciones preventivas propuestas a lo largo del proceso de aseguramiento de la calidad, que describiremos posteriormente en la subsección 15.6.2 en la página siguiente.

El Director del Proyecto será el que proponga los cambios, las acciones correctivas y preventivas que han de ser aprobadas o rechazadas por el cliente.

A lo largo de todo este proceso no se debe perder de vista el alcance del proyecto para contrastar los cambios que llegasen a surgir. Durante el desarrollo del proyecto, se puede necesitar una revisión o redefinición del alcance para reflejar estos cambios.

15.5. Gestión del riesgo

A lo largo del ciclo de vida del proyecto pueden surgir situaciones que afecten a los objetivos del proyecto, tanto positiva, como negativamente. En el primer caso, se trataría de oportunidades que habría que aprovechar, mientras que en el segundo, estaríamos ante riesgos, que habría que manejar. La Figura 15.2 muestra los tipos de riesgos a los que vamos a tener que hacer frente a lo largo del proyecto.

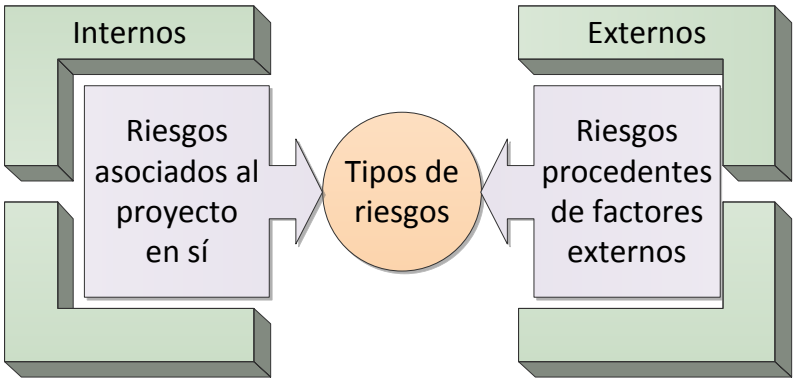


Figura 15.2: Tipos de riesgos.

Las tareas a llevar a cabo durante la gestión del riesgo serán las siguientes:

1. Inicialmente identificaremos todos los posibles riesgos y los catalogaremos, pudiendo ser actualizados en cualquier punto del proyecto.

2. Asignaremos prioridades a las distintas categorías de riesgos establecidas, en función del posible impacto que puedan tener en el proyecto.

3. Ante una causa de riesgo, se trataría de actuar conforme a las acciones que se exponen a continuación:

   a) Eliminar la causa del riesgo: es la mejor opción, pero no siempre es posible.

   b) Reducir su impacto, mitigando su magnitud o probabilidad.

Cuando no se pueda resolver mediante la primera acción expuesta, bajaremos al siguiente nivel, resolviendo la causa del riesgo en la medida que su naturaleza nos permita.

15.6. Gestión de la calidad

La gestión de la calidad de un proyecto incluye todos aquellos procesos y actividades que tienen por finalidad determinar las políticas de calidad, los objetivos y responsabilidades de tal forma que el proyecto satisfaga las necesidades por las que se ha asumido su desarrollo. La gestión de la calidad se realizará a lo largo de todas las fases que conforman el proyecto, subdividiéndose en los siguientes procesos/actividades.

15.6.1. Planificación de la calidad

El primer paso en la gestión de la calidad es identificar los estándares de calidad que serán usados en el proyecto. En la planificación de la calidad, partiendo del alcance del proyecto, se identificarán los requisitos de calidad del mismo, documentando cómo el proyecto va a cumplir dichos requisitos.

15.6.2. Realizar aseguramiento de la calidad

Es el proceso por el que se revisan los requisitos de calidad y los resultados de las medidas del control de calidad, para asegurar que se están usando los estándares de calidad apropiados. La herramienta que usaremos para realizar el aseguramiento de calidad serán auditorías internas cuyos objetivos serán los siguientes:

- Identificar las buenas prácticas que se estén implementando.

- Identificar los defectos/huecos.

- Compartir las buenas prácticas introducidas o implementadas en proyectos similares.

- Ofrecer ayuda para mejorar la implementación de los procesos obteniendo un incremento de la productividad del equipo.

Puesto que no se trata de un proyecto de grandes dimensiones, dichas auditorías de calidad serán una parte de las reuniones de seguimiento. Las salidas que obtendremos serán las siguientes:

- Solicitudes de cambio: constituyen la entrada del proceso de gestión del cambio descrito en la sección 15.4 en la página 201.

- Acciones correctivas o preventivas.

- Actualización de la planificación del proyecto: como consecuencia de este proceso, pueden sufrir cambios los procesos de planificación de la calidad, de planificación del proyecto, de planificación de costes, etc.

15.6.3. Realizar control de calidad

El control de calidad es el proceso por el cual se monitorizan y registran los resultados de ejecutar las actividades de calidad para asegurar el rendimiento y recomendar los cambios necesarios. Por esta razón, se realizará este proceso a lo largo de toda la vida del proyecto.

15.7. Estándares de codificación

Para el desarrollo de la aplicación se ha utilizado el estándar de codificación ANSI C para el que se seguirán las convenciones habituales de codificación de este lenguaje relativas al nombre de variables, constantes, estructuras, uso de llaves, etc.

15.8. Planificación y presupuesto

Para terminar con el plan de proyecto, se va a incluir en primer lugar, la planificación de todas las fases descritas anteriormente a excepción del estudio de viabilidad. Esta fase no debe ser incluida en la planificación puesto que se trata de un estudio para analizar si llevar a cabo el proyecto va a resultar viable. A continuación, presentaremos el presupuesto descompuesto en sus partes principales.

15.8.1. Secuenciación de tareas

La fecha de inicio del proyecto se establece el día 20 de Septiembre de 2010 y la fecha de finalización en base a la actual planificación se emplaza el día 7 de Julio de 2011. Entre estas dos fechas se sitúa el desarrollo del proyecto con un total de 1.666 horas. El reparto de horas para cada una de las tareas del proyecto se presenta en la Tabla 15.1.

Tabla 15.1: Uso de Tareas durante el Proyecto

Tarea		Horas
1	Planificación	35
2	Estado del Arte	210
3	Análisis	189
4	Diseño	154
5	Implementación	413
6	Pruebas	525
7	Implantación	42
8	Redacción del Proyecto	616
	Total	**1.666**

El diagrama de Gantt de la Figura 15.3 en la página siguiente, muestra la sucesión de cada una de las tareas que componen el desarrollo del proyecto.

Además, en la Figura 15.4 en la página 206, se puede observar la planificación de las reuniones de seguimiento y las fechas de las diferentes entregas de hitos fijadas.

15.8.2. Estimación de los recursos humanos

Teniendo en cuenta la planificación de las tareas descritas en el apartado anterior, procedemos a exponer en la Tabla 15.2 en la página 206 el coste asociado a los recursos humanos utilizados a lo largo de toda la vida del proyecto. Para el cálculo de la dedicación expresada en hombres mes, hemos supuesto jornadas de 7 horas/día, semanas de 5 días y meses de 4,2 semanas, por lo que:

$$1\,Hombre\,mes = 147\,horas = 7\,\frac{horas}{día} \times 5\,días \times 4,2\,semanas$$

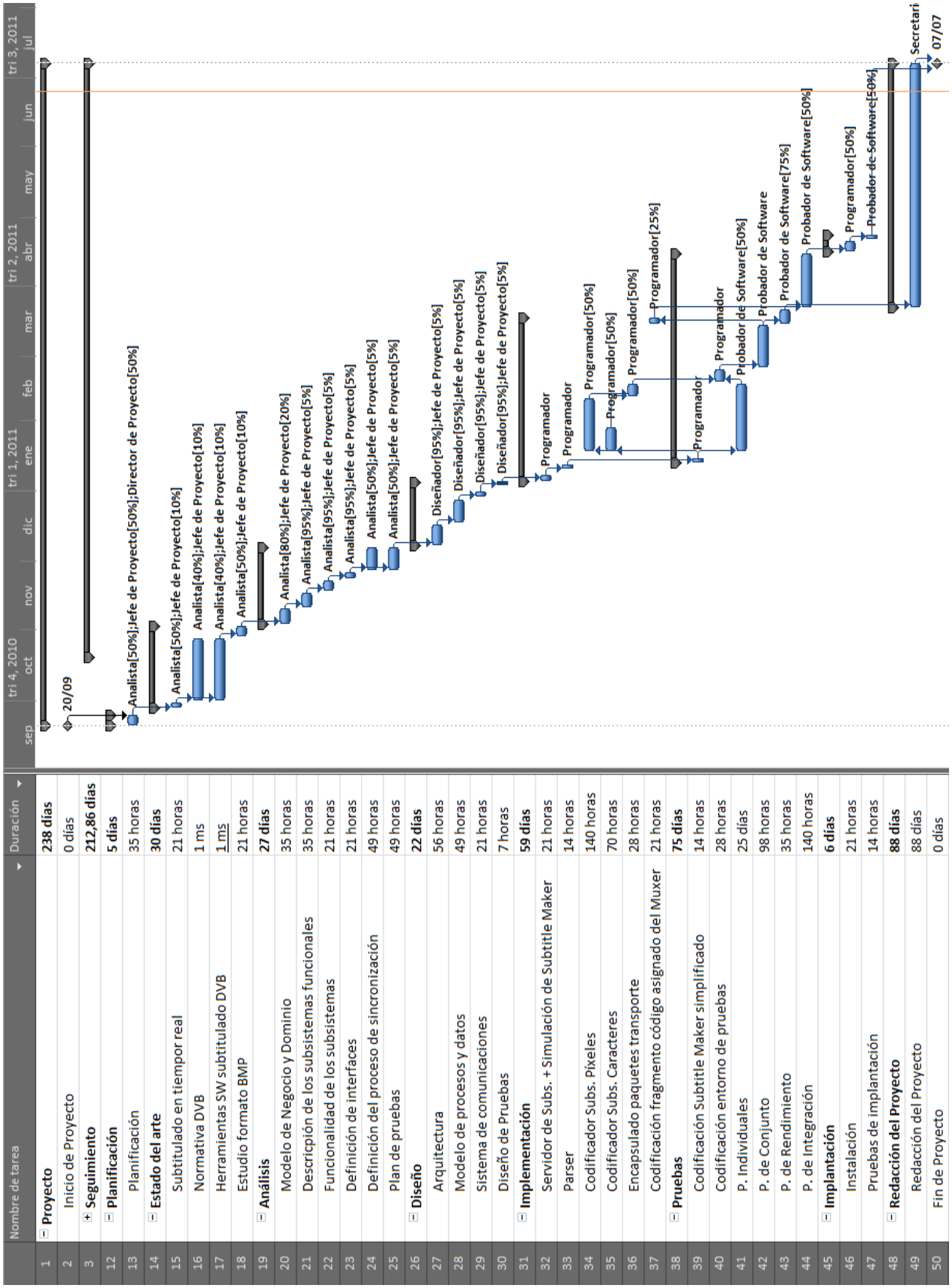


Figura 15.3: Diagrama de Gantt.

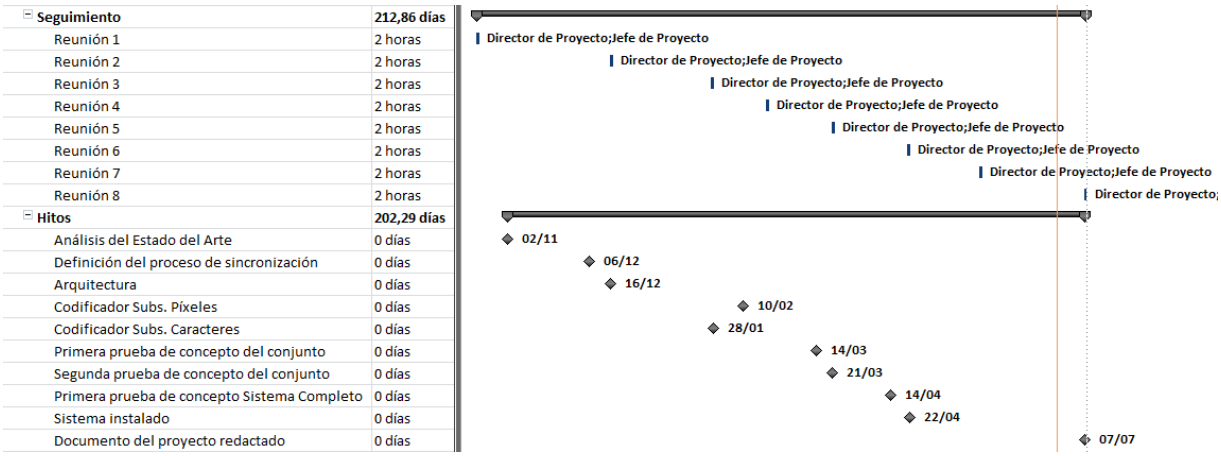


Figura 15.4: Reuniones de seguimiento e hitos.

Tabla 15.2: Costes Recursos Humanos

Categoría	Dedicación (Horas)	Dedicación (hombres mes)	Coste hombre mes	Coste (Euro)
Director de Proyecto	33,5	0,23	16.611,00 €	3.785,50 €
Jefe de Proyecto	90,9	0,62	11.025,00 €	6.817,50 €
Analista	311,85	2,12	9.261,00 €	19.646,55 €
Diseñador	126,35	0,86	7.350,00 €	6.317,50 €
Programador	211,75	1,44	5.880,00 €	8.470,00 €
Probador de Software	288,75	1,96	3.675,00 €	7.218,75 €
Secretario	308	2,10	2.205,00 €	4.620,00 €
	Hombres mes	9,327210884	**Total**	**56.875,80 €**

15.8.3. Estimación de los recursos materiales

A los costes derivados del uso de recursos humanos hay que añadir los que tienen su origen en el uso de recursos materiales. Para realizar una estimación de su coste vamos a establecer un periodo de amortización de 8 años para los elementos de tratamiento de información y de 6 años para los programas informáticos. En base a esta información, el coste asociado a cada elemento viene determinado por la siguiente expresión:

$$Coste = \frac{A}{B} \times C \times D \tag{15.1}$$

Donde:

1. A = nº de meses desde la fecha de facturación en que el equipo es utilizado.
2. B = periodo de depreciación.
3. C = coste del equipo (sin IVA).
4. D = % del uso que se dedica al proyecto. Lo fijamos a 100 %.

La Tabla 15.3 refleja el desglose de los costes asociados a los recursos materiales.

Tabla 15.3: Costes Recursos Materiales

Concepto	Coste	Dedicación (meses)	Periodo de depreciación	Coste imputable
PC Sobremesa I7	1.600,00 €	10	96	166,67 €
Buy Best Easy Home TDT Black	29,90 €	2,5	96	0,78 €
Energy System T3250	29,90 €	2,5	96	0,78 €
Philips DTR220	53,90 €	2,5	96	1,40 €
Daewoo DSD-800M	29,90 €	2,5	96	0,78 €
MxOnda Mx-STB5289	33,00 €	2,5	96	0,86 €
Lauson 1025N	40,00 €	2,5	96	1,04 €
Gigaset M280 T EPG	43,95 €	2,5	96	1,14 €
Boston DTT 4100	34,35 €	2,5	96	0,89 €
i-Joy i-Vision TDT	49,95 €	2,5	96	1,30 €
AXIL RT160	31,90 €	2,5	96	0,83 €
LabMU	19.650 €	2,5	72	682,29 €
Microsoft Windows 7 Professional	309,00 €	3	72	12,88 €
Microsoft Project 2010	1.300,00 €	0,25	72	4,51 €
Micrsoft Visio 2010	725,00 €	3	72	30,21 €
Microsoft Office 2010	699,00 €	0,25	72	2,43 €
Ubuntu 11.04	0,00 €	10	72	0,00 €
Gimp 2.6	0,00 €	3	72	0,00 €
Geany 0.20	0,00 €	4,5	72	0,00 €
			Total	**908,79 €**

A estos costes debemos añadir otros costes directos del proyecto, como son los fungibles, valorados en 30,00 €.

15.8.4. Costes totales

El presupuesto total de este proyecto asciende a la cantidad de CIENTO NUEVE MIL CUATROCIENTOS TREINTA Y NUEVE EUROS CON NOVENTA Y CINCO CÉNTIMOS. De éstos, 107.769,96 € corresponden al desarrollo completo del Proyecto y 1.669,99 € a los costes soportados por el cliente, derivados de la implantación del sistema en sus instalaciones. Las siguientes tablas recogen el desglose de los costes.

Tabla 15.4: Costes del Proyecto

Concepto	Valor
Recursos Humanos	56.875,80 €
Recursos Materiales	908,79 €
Otros costes directos (Fungibles)	30 €
Gastos Generales (20 % Recursos Humanos)	11.375,16 €
Subtotal 1	**69.189,75 €**
Beneficios Empresariales (20 % Subtotal)	13.837,95 €
Subtotal 2	**83.027,70 €**
Costes del Riesgo (10 % Subtotal 2)	8.302,77 €
Base Imponible	**91.330,47 €**
I.V.A. (18 % Base Imponible)	16.439,49 €
TOTAL	**107.769,96 €**

Tabla 15.5: Costes del Cliente

Concepto	Precio/Unidad	Unidades	Costes
PC Sobremesa I7	1.600,00 €	1	1.600,00 €
WinTV-NOVA-TD-500 HD	69,99 €	1	69,99 €
	TOTAL (I.V.A Incluido)		**1.669,99 €**

Parte V

Conclusiones y Líneas futuras

Capítulo 16

Conclusiones

En el presente Proyecto de Fin de Carrera se ha abordado el problema del subtitulado sincronizado de programas de televisión en directo. Como se ha visto, el sistema desarrollado plantea una solución a este problema mediante la codificación de subtítulos DVB-SUB en las dos vertientes que se plantean en la norma ETSI EN 300 743 [16]: codificación basada en píxeles y codificación basada en cadenas de caracteres, en ambos casos a partir de subtítulos creados mediante tecnología de reconocimiento del habla, que permiten obtener la información temporal necesaria para la posterior sincronización entre audio y subtítulos. Tras un intenso estudio del estado del arte y tomar la decisión de la alternativa de solución más adecuada para cumplir con los requisitos del sistema, se planteó una arquitectura de sistema en la que se contemplaban ambos tipos de codificación, en vez de diseñar dos arquitecturas diferentes para cada tipo.

Por esta razón, se planteó un sistema modular que facilitase la simplicidad y reutilización de código, de forma que, en aquellos puntos de la ejecución en que la tarea a realizar fuera común en ambas partes, el sistema únicamente tuviera que hacer uso del módulo encargado de dicha tarea. Además, para cada uno de los tipos de codificación implementados, se incorporó una fase de mejoras que permitieran obtener un sistema con las mejores prestaciones posibles, siempre teniendo en cuenta los requisitos y la funcionalidad del sistema especificados.

El diseño de la arquitectura realizado es uno de los aspectos que, a la postre, han resultado claves por facilitar no sólo la comunicación entre módulos, sino también, por simplificar en gran medida la a priori, tarea más costosa del proyecto: la sincronización de los subtítulos respecto al audio y el vídeo.

16.1. Conclusiones de las etapas comunes

Las etapas comunes se encuentran a la entrada y a la salida del sistema. La mejor forma de concluir una valoración sobre las mismas es considerar si se ajustan

a las especificaciones de los requisitos en cuanto al tipo de comunicación que deben asumir y al contenido que deben ser capaces de recibir o emitir.

En el caso de la entrada, se ha diseñado un Servidor TCP para realizar la gestión de los subtítulos en formato TTML, los cuales son interpretados por el Parser extrayendo la información útil de los mismos. Dicha información, es enviada al módulo de codificación que proceda, dependiendo del tipo de codificación a realizar. Siguiendo los requisitos identificados como UR-005 y UR-006 (ver sección 8.1 en la página 125), en cualquiera de los dos casos posibles, la forma establecida para enviar la información ha sido a través de una cola de sistema con destino al Muxer, por la que se envía la información codificada en paquetes de transporte de 188 bytes, es decir, tal y como se especifica en los requisitos mencionados. Podemos concluir que ésta es una elección adecuada puesto que permite independizar muy fácilmente la implementación de cada módulo del interfaz entre ellos, y permite la evolución de manera independiente.

16.2. Conclusiones de la codificación basada en píxeles

Sin lugar a dudas, ésta ha resultado la etapa del proyecto más laboriosa debido a la dificultad que ha entrañado implementar la codificación RLC indicada por la norma y a la imposibilidad de realizar pruebas sistemáticas libres de errores, tras las primeras versiones de código realizadas. Esto nos llevó a detener el avance del proyecto para diseñar un banco de pruebas fiable que permitiese optimizar el tiempo durante las pruebas y que facilitase la localización de los errores cometidos.

Una vez superados estos obstáculos, toda nuestra atención se volvió a centrar en proseguir con la codificación de los subtítulos hasta que finalmente pudimos visualizar en pantalla una imagen que se correspondiese con el subtítulo que se pretendía codificar y que se ajustase a las especificaciones, es decir, dos líneas de subtítulos centrados sobre fondo transparente, con los tamaños y color de letra indicados en los requisitos del sistema.

Además, una vez alcanzados los requisitos definidos, se realizaron una serie de mejoras que permitieron obtener subtítulos en formato imagen mucho más nítidos y eficaces en cuanto al ancho de banda requerido en su emisión.

16.3. Conclusiones de la codificación basada en cadenas de caracteres

Por el contrario, esta codificación ha resultado mucho más sencilla que la basada en píxeles, debido a la simplicidad que implica no tener que tratar con imágenes. Sin

embargo, mientras que en la otra codificación resultó muy sencillo lograr subtítulos centrados en pantalla, en ésta supuso el mayor obstáculo a superar, obligando a desarrollar un algoritmo para tal propósito.

Tal y como se indicó anteriormente, este tipo de codificación está muy poco extendida entre los fabricantes de receptores TDT por lo que se desaconseja su uso dentro de un sistema de subtitulado para televisión. A pesar de ello, para cumplir con los requisitos del sistema, se implementó una solución que cumpliera todas las especificaciones.

16.4. Conclusiones generales

Tras unificar todas las partes desarrolladas durante el proyecto, únicamente quedaba dotar al sistema de capacidad para preparar un flujo de subtítulos sincronizado respecto al contenido y así cumplir con uno de los principales requisitos establecidos. La forma de producir dicha sincronización, como consecuencia del diseño realizado, resultó ser la misma en ambas codificaciones y además, muy sencilla de lograr según el procedimiento explicado en la sección 9.3 en la página 137. Esto se debe principalmente a la definición de interfaces y comportamiento dinámico del sistema realizado como parte de este proyecto , pues el hecho de que la sincronización se pueda lograr de esta forma, tiene su origen en la distribución de la funcionalidad y en el conjunto de interfaces desarrollados.

El principal objetivo de este proyecto consistía en demostrar la viabilidad de un sistema de subtitulado de programas de TV en directo, en el que los subtítulos estuvieran sincronizados con los contenidos del programa. Después de las pruebas realizadas tanto a nivel individual del sistema, como de conjunto, dentro del SSAS, se concluye que la sincronización es posible, dentro de las limitaciones que actualmente existen en sistemas de este tipo y que comentamos al final del capítulo de pruebas.

Es conveniente resaltar el hecho de que la mejora de los sistemas ASR influirá notablemente en el rendimiento del SSAS, puesto que en escenarios multi-locutor, con ruido, música, etc., bajan mucho las prestaciones de la tasa de reconocimiento.

Lo único que necesita el sistema para lograr una sincronización adecuada en todas las situaciones, es que las marcas temporales derivadas de la transcripción de voz a texto, se mantengan dentro de un margen de error inferior a 1 segundo y no sufran errores como consecuencia de esos escenarios actualmente no controlados. Estas marcas temporales, en muchas ocasiones sufren errores, por no hablar de la calidad de las transcripciones, pero en el momento en que la tecnología ASR mejore en esos y otros aspectos, el sistema desarrollado podrá generar subtítulos sincronizados ante cualquier situación y de buena calidad.

Esto además no supondrá apenas cambios en el sistema, puesto que como ya se ha dicho, es completamente modular, por lo que se ha dejado un sistema preparado

para acoger esta posibilidad en futuros desarrollos.

En conclusión, el grado de cumplimiento de los requisitos planteados al inicio puede considerarse alcanzado y en muchos casos superado, consiguiendo en algunos aspectos una mejora de las prestaciones que se suponían del sistema.

Finalmente, mencionar la contribución realizada a las Jornadas de Telecom I+D [54] mediante el envío del artículo denominado *Subtitulado automático sincronizado DVB* [53] que, tras el proceso de evaluación y revisión correspondiente, ha sido aceptado para su presentación en la sesión de posters del congreso.

Capítulo 17

Líneas Futuras

17.1. Introducción

En este capítulo vamos a abordar algunas de las líneas futuras que no han sido incluidas dentro del ámbito del proyecto, ya sea por su naturaleza, o por exceder los límites de lo que supone un proyecto de fin de carrera.

17.2. MPEG-2/DVB++

El concepto de MPEG-2/DVB++ [55] va más allá de lo que hemos visto hasta ahora. Este sistema propone una forma de realizar la codificación de los subtítulos por la cual se divide el proceso de subtitulado en dos partes, cada una de las cuales, se realiza en dos puntos distintos de la comunicación:

1. Codificación de los subtítulos: Este proceso, al igual que en el sistema planteado en este proyecto, se realiza en origen, es decir, se realiza durante la emisión de los contenidos y consiste en generar el flujo de subtítulos. Sin embargo, este flujo presenta un ligero matiz respecto al generado hasta ahora: no está sincronizado sino que incorpora una serie de marcas temporales que nos permiten lograr la sincronización de los subtítulos respecto al vídeo y al audio en recepción.

2. Sincronización de los subtítulos: A diferencia de lo planteado hasta el momento, este proceso se realiza durante la decodificación de los flujos, es decir, en el STB receptor. Para ello, no sólo se tiene en cuenta la información de las tablas PSI/SI para reconstruir la señal de televisión, sino que el receptor debe tener en cuenta las marcas introducidas para conseguir sincronizar los subtítulos.

Mediante este sistema, vamos a dar la opción al usuario de que seleccione la versión en directo o la subtitulada sincronizada haciendo uso de un receptor adaptado al formato MPEG-2/DVB++. Esta es la forma óptima ya que no requiere ancho de

banda adicional para transmitir la señal TDT pero limita la versión sincronizada a aquellos usuarios que dispongan de un receptor adaptado. Esta es la principal razón por la cual se descartó este mecanismo, además de que el problema del ancho de banda desaparece al tratarse de un canal IPTV.

Un sistema como el planteado en este punto daría lugar igualmente a una versión subtitulada sincronizada y retrasada respecto a la original, ya que, aunque la sincronización se realice en recepción, el sistema receptor debe analizar las marcas temporales derivadas del formato MPEG-2/DVB++ y realizar la sincronización.

17.3. Subtítulos en Teletexto

A pesar de que los subtítulos codificados en este formato se encuentran atrasados tecnológicamente respecto a los codificados en DVB-SUB y que la tendencia en las cadenas de televisión debería ser la adopción del sistema de subtitulado DVB-SUB, la realidad actual es que el teletexto se encuentra aún muy extendido entre las cadenas, siendo en algunas de ellas, el único mecanismo habilitado para visualizar los subtítulos. En vistas de esto, la inclusión de este tipo de codificación en el proyecto, como una alternativa seleccionable por el usuario, lo haría mucho más completo y llamativo, puesto que cubriría a la totalidad del público usuario de subtítulos en televisión.

17.4. Subtítulos multi-interlocutor simultáneos

Debido a las limitaciones impuestas por el reconocedor de voz, cada vez que se genera un subtítulo se asume que éste se refiere a un sólo interlocutor, y en consecuencia, todo el subtítulo que se genera, presenta el mismo color. No obstante, en una situación real, es posible que varios interlocutores hablen a la vez y que, por consiguiente, cada renglón de subtítulo deba aparecer en un color diferente para identificar al interlocutor asociado a esa frase. En un futuro, suponiendo que la tecnología mejore de tal forma que se pueda diferenciar más de un interlocutor de manera simultánea, sería recomendable incorporar al código la posibilidad de asignar colores diferentes a cada línea de subtítulo. Dada la arquitectura actual del sistema, esto implicaría cambios no sólo en el sistema aquí desarrollado, sino también en el Subtitle Maker, pues éste sería el encargado de indicar el color de cada subtítulo formateado en TTML.

17.5. Subtítulos en Alta Definición

Aunque ya hemos mencionado algo sobre los subtítulos en alta definición, el ámbito de este proyecto no cubre la generación de subtítulos con este formato de

salida. No obstante, ante la incipiente demanda de contenidos HD, cualquier sistema de subtitulado que pretenda brindar soporte en un futuro muy cercano, deberá ofrecer la posibilidad de emitir los subtítulos en alta definición, o de lo contrario, será un sistema obsoleto, incluso antes de salir a la luz. El sistema desarrollado es fácilmente modificable en este sentido, por lo que extender su funcionalidad para contemplar esta posibilidad no debería ser un problema.

17.6. Emisión de subtítulos DVB++ a través de HbbTV

HbbTV es una plataforma de emisión de contenidos bajo demanda que combina los servicios de radiodifusión y banda ancha, siendo también conocida como Televisión Híbrida. Ésta y otras formas de TV híbrida con acceso a la red broadband (banda ancha) constituyen el futuro inminente de la televisión por lo que sería importante contemplar esta línea de investigación en futuros proyectos.

Parte VI

Glosario y Bibliografía

Glosario

ASR Automatic Speech Recognition, página 16

ATIS Automatic Terminal Information Service, página 20

BAT Bouquet Association Table, página 46

BMP Bit Mapped Picture, página 103

CAT Conditional Access Table, página 39

CDS CLUT definition segment, página 54

CLI Command Line Interface, página 89

CLUT Colour Look-UP Table, página 24

DDS Display definition segment, página 53

DIB Device Independent Bitmap, página 103

DIT Discontinuity Information Table, página 48

DVB Digital Video Broadcasting, página 10

EIT Event information table, página 47

EODSS End of display set segment, página 54

HbbTV Hybrid Broadcast Broadband TV, página 4

HMM Hidden Markov Model, página 17

HTTP Hypertext Transfer Protocol, página 90

IRD Integrated Receiver Decoder, página 24

NIT Network Information Table, página 39

ODS Object data segment, página 54

PAT Program Association Table, página 39

PCR Program Clock Reference, página 27

PCS page composition segment, página 51

PES Packetized elementary stream, página 25

PMT Program Map Table, página 39

POF Probabilistic Optimal Filtrering, página 17

PS Program Stream, página 23

PSI Program Specific Information, página 34

RCS region composition segment, página 51

RGB Red Green Blue, página 105

RLC run-length coding, página 52

RLE Run Length Encoding, página 105

RNT RAR Notification Table, página 233

RST Running status table, página 47

RTP Real Time Protocol, página 90

RTSP Real Time Streaming Protocol, página 90

SCR System Clock Reference, página 26

SDT Service description table, página 47

SIT Selection Information Table, página 48

SNR Signal to Noise Ratio, página 19

SSAS Sistema de subtitulado automático sincronizado, página 129

ST Stuffing table, página 48

STB Set-top Box, página 179

STFT Short-time Fourier Transform, página 17

TDT Time/date table, página 48

TOT Time offset table, página 48

TS Transport Stream, página 23

TSDT Transport Stream Description Table, página 39

TTS Text To Speech, página 112

UHF Ultra High Frequency, página 23

VBI Vertical Blanking Interval, página 76

Bibliografía

[1] España. Ley 7/2010, de 31 de marzo, General de la Comunicación Audiovisual. Boletín Oficial del Estado, 1 de abril de 2010, núm. 79, p. 30157.

[2] A.J. Mason. and R.A. Salmon, "Factors affecting perception of audio-video synchronisation in television" (2009). BBC R&D White Paper WHP176, 2009

[3] F. Jurcicek, "Speech recognition for live TV captioning" (2009). IEEE Communication Society, SLTC Newsletter, April 2009

[4] Garcia, J.E.; Ortega, A.; Lleida, E.; Lozano, T.; Bernues, E.; Sanchez, D. (2009). "Audio and text synchronization for TV news subtitling based on Automatic Speech Recognition," IEEE International Symposium on Broadband Multimedia Systems and Broadcasting, 2009. . , vol., no., pp.1-6, 13-15 May 2009

[5] Centro Español de Subtitulado y Audiodescripción. 2011. Disponible [en línea]: http://www.cesya.es [28 de marzo de 2011]

[6] Encuesta de Discapacidad, Autonomía personal y situaciones de Dependencia (EDAD). Instituto Nacional de Estadística, 2008. 12p. Disponible [en línea]: http://www.ine.es/prensa/np524.pdf [29 de marzo de 2011]

[7] Encuesta Nacional de Inmigrantes 2007: una monografía. [Madrid]: Instituto Nacional de Estadística, 2007. 160p. Disponible [en línea]: http://www.ine.es/prodyser/pubweb/eni07/eni07.pdf [29 de marzo de 2011]

[8] DE CASTRO, Mercedes. *"Introducción al subtitulado en tiempo real"* (Marzo 2011).

[9] LÓPEZ, J. Francisco. *"Herramientas propietarias y gratuitas de subtitulado"*. Máster en Tecnologias de Apoyo, Accesibilidad y diseño para todos, TA^{2}DIS (2011, Leganés, Madrid).

[10] DE CASTRO, Mercedes; JIMÉNEZ, Javier; RUIZ, Belén. "*SINCRONIZACIÓN ENTRE SUBTÍTULOS Y AUDIO EN LA TELEVISIÓN EN DIRECTO*". En: *AMADIS 2010*, (Madrid 18-19 de noviembre de 2010). 5a ed.

[11] Dragon NaturallySpeaking 11. 2011. Disponible [en línea]: http://www.nuance.es/naturallyspeaking/products/premium.asp [28 de marzo de 2011]

[12] Anglatècnic Fingertext. Ingeniería y Desarrollo de Sistemas para los sectores Broadcast y Tecnologías de la Información. Disponible [en línea]: http://www.anglatecnic.com/es-6-Fingertext- %28Sistema-Teletexto %29.html [28 de marzo de 2011]

[13] DVB. Disponible [en línea]: http://www.dvb.org/ [11 de abril de 2011]

[14] ISO/IEC. Generic coding of moving pictures and associated audio information: Systems. ISO/IEC 13818-1:2007. 3a ed. Suiza: 2007.

[15] ETSI. Digital Video Broadcasting (DVB); Specification for Service Information (SI) in DVB systems. ETSI EN 300 468. Francia: 2010.

[16] ETSI. Digital Video Broadcasting (DVB); Subtitling Systems. ETSI EN 300 743. Francia: 2006.

[17] ETSI. Digital Video Broadcasting (DVB); Specification for conveying ITU-R System B Teletext in DVB bitstreams. ETSI EN 300 472. Francia: 2003.

[18] Project-X - DVB demux Tool. Disponible [en línea]: http://project-x.sourceforge.net/ [9 de septiembre de 2010]

[19] ProjectX reference. Disponible [en línea]: http://spanish.doom9.org/index.html?/DigiTV/projectx.htm [4 de abril de 2011]

[20] VideoLAN's Wiki. Disponible [en línea]: http://wiki.videolan.org/Main_Page [19 de julio de 2010]

[21] VLC Features Formats. Disponible [en línea]. http://wiki.videolan.org/VLC_Features_Formats [4 de abril de 2011]

[22] VLC command-line help. Disponible [en línea]. http://wiki.videolan.org/VLC_command-line_help [20 de julio de 2010]

[23] FFmpeg. Disponible [en línea]: http://ffmpeg.org/ [28 de julio de 2010]

[24] FFmpeg License and Legal Considerations. Disponible [en línea]: http://www.ffmpeg.org/legal.html [5 de abril de 2011]

[25] FFmpeg General Documentation. Supported File Formats and Codecs. Disponible [en línea]: http://ffmpeg.org/general.html#TOC3 [28 de julio de 2010]

[26] About FFmpeg. Disponible [en línea]: http://ffmpeg.org/about.html [5 de abril de 2011]

[27] FFmpeg-Based Projects. Disponible [en línea]: http://ffmpeg.org/projects.html [5 de abril de 2011]

[28] FFmpeg Documentation. Disponible [en línea]: http://www.ffmpeg.org/ffmpeg.html [28 de julio de 2010]

[29] FFserver Documentation. Disponible [en línea]: http://www.ffmpeg.org/ffserver.html [6 de abril de 2011]

[30] FFplay Documentation. Disponible [en línea]: http://ffmpeg.org/ffplay.html [29 de julio de 2010]

[31] Libavfilter Documentation. Disponible [en línea]: http://ffmpeg.org/libavfilter.html [7 de abril de 2011]

[32] Captioning and Subtitling Software. Disponible [en línea]: http://www.softelgroup.com/Our-Solutions/Subtitling-and-Captioning [8 de abril de 2011]

[33] Softel subtitling, captioning and ancillary data global customers. Disponible [en línea]: http://www.softelgroup.com/Our-Customers/subcust [8 de abril de 2011]

[34] Softel Swift Create. Disponible [en línea]: http://www.softelgroup.com/Our-Products/Swift-Create [13 de julio de 2010]

[35] Softel Swift TX. Disponible [en línea]: http://www.softelgroup.com/Our-Products/Swift [13 de julio de 2010]

[36] Softel ScheduleSmart™. Disponible [en línea]: http://www.softelgroup.com/Our-Products/ScheduleSmart [13 de julio de 2010]

[37] FAB Subtitler Live Edition. Disponible [en línea]: http://www.fab-online.com/eng/subtitling/broadcast/subtlive.htm [28 de marzo de 2011]

[38] Cavena Tempo - subtitle preparation system. Disponible [en línea]: http://www.cavena.com/Subtitle-preparation/Tempo-subtitle-preparation-system/ [10 de abril de 2011]

[39] Cavena STU - Subtitle Transmission Unit. Disponible [en línea]: http://www.cavena.com/Subtitle-Transmission/Subtitle-transmission-unit/ [10 de abril de 2011]

[40] Cavena STC - Subtitle Transmission Control. Disponible [en línea]: http://www.cavena.com/Subtitle-Transmission/Subtitle-Transmission-Control/ [10 de abril de 2011]

[41] Cavena. Subtitle Transmission formats. Disponible [en línea]: http://www.cavena.com/Subtitle-Solutions/Subtitle-Transmission-formats/ [10 de abril de 2011]

[42] SysMedia Subtitling. Disponible [en línea]: http://www.sysmedia.com/solutions/subtitling_solutions.asp [10 de abril de 2011]

[43] ETSI. Digital Video Broadcasting (DVB); Subtitling Systems. ETSI EN 300 743. Francia: 2006.MIANO, John. *Compressed image file formats: JPEG, PNG, GIF, XBM, BMP*. 1a ed. Reading, Massachusetts: Addison Wesley, 1999. 264 p. ISBN: 0-201-60443-4.

[44] Microsoft. *"DIB y sus usos"*. Versión 2.5, 2005. Disponible [en línea]: http://support.microsoft.com/kb/q81498/ [29 de marzo de 2011]

[45] APEINTA. Apuesta por la enseñanza inclusiva. Disponible [en línea]: http://www.apeinta.es/apeinta/ [20 de mayo de 2011]

[46] DE CASTRO, Mercedes, et al. "Hermes-TDT. Medición de emisión y audiencia de los servicios de accesibilidad en la TDT". En: *Telecom I+D 2009*, (Madrid 24-26 de noviembre de 2009).

[47] Timed Text Markup Language (TTML) 1.0. W3C Recommendation 18 November 2010. Disponible [en línea]: http://www.w3.org/TR/ttaf1-dfxp/ [27 de enero de 2011]

[48] Tarjeta sintonizadora WinTV-NOVA-TD 500 HD. Disponible [en línea]: http://www.hauppauge.es/site/products/data_novat500.html [16 de diciembre de 2010]

[49] GD module for php5. Disponible [en línea]: http://packages.debian.org/sid/php5-gd [19 de enero de 2011]

[50] UTF-8 encoding table and Unicode characters. Disponible [en línea]: http://www.utf8-chartable.de/ [9 de diciembre de 2010]

[51] Multiuser Digital TV Lab (LabMU). Disponible [en línea]: http://www.xpertiasi.com/index.php/en/labmu [15 de Marzo de 2011]

[52] AENOR. Subtitulado para personas sordas y personas con discapacidad auditiva. UNE 153010. Madrid: AENOR, 2003.

[53] LORENZO PRADA, Alejandro, et al. "Subtitulado automático sincronizado DVB". En: XXI Jornadas TELECOM I+D, (28-30 de septiembre de 2011). Aceptado para su presentación en la sesión de posters.

[54] XXI Jornadas Telecom I+D. Las TIC en las ciudades del siglo XXI: la Smart City. Disponible [en línea]: http://www.telecom-id.com/index.php?st=cfp [11 de Mayo de 2011]

[55] DE CASTRO, Mercedes, et al. "Synchronization between subtitles, audio and video in digital television live subtitling".

Parte VII

Anexos

Apéndice A

Valores de PID permitidos

Tabla A.1: Valores de PID permitidos [15].

Valor en Hexadecimal	Descripción
0x0000	Program Association Table (PAT)
0x0001	Conditional Access Table (CAT)
0x0002	Transport Stream Description Table (TSDT)
0x0003 - 0x000F	Reservado
0x0010	Network information table (NIT) y Stuffing table (ST)
0x0011	Service description table (SDT), Bouquet information table (BAT) y Stuffing table (ST)
0x0012	Event information table (EIT) y Stuffing table (ST)
0x0013	Running status table (RST) y Stuffing table (ST)
0x0014	Time/date table (TDT), Time offset table (TOT) y Stuffing table (ST)
0x0015	Sincronización de red
0x0016	RAR Notification Table (RNT)
0x0017 - 0x001B	Reservado para usos futuros
0x001C	Señalización en banda
0x001D	Medición
0x001E	Discontinuity Information Table (DIT)
0x001F	Selection Information Table (SIT)
0x0020 - 0x1FFE	Datos de vídeo, audio y privados
0x1FFF	Paquetes no válidos

Apéndice B

Valores posibles del campo *table_id*

Tabla B.1: Asignación de valores a table_id [14].

Valor en Hexadecimal	Descripción
0x00	program_association_section
0x01	conditional_access_section (CA_section)
0x02	TS_program_map_section
0x03	TS_description_section
0x04	ISO_IEC_14496_scene_description_section
0x05	ISO_IEC_14496_object_descriptor_section
0x06 - 0x37	ITU-T Rec. H.222.0 | ISO/IEC 13818-1 reserved
0x38 - 0x3F	Defined in ISO/IEC 13818-6
0x40 - 0xFE	Privado del usuario
0xFF	Prohibido

Apéndice C

Tipos de Subtítulos

Tabla C.1: Tipos de subtitling_type del descriptor de subtítulos [15].

Subtitling type	Descripción
0x00	Reservado para uso futuro
0x01	Subtítulos EBU Teletext
0x02	EBU Teletext asociado
0x03	Datos VBI
0x04 - 0x0F	Reservado para uso futuro
0x10	Subtítulos DVB (normal) sin relación de aspecto específica
0x11	Subtítulos DVB (normal) para relación de aspecto 4:3
0x12	Subtítulos DVB (normal) para relación de aspecto 16:9
0x13	Subtítulos DVB (normal) para relación de aspecto 2.21:1
0x14	Subtítulos DVB (normal) para mostrar en alta definición
0x15 - 0x1F	Reservado para uso futuro
0x20	Subtítulos DVB (para sordos) sin relación de aspecto específica
0x21	Subtítulos DVB (para sordos) para relación de aspecto 4:3
0x22	Subtítulos DVB (para sordos) para relación de aspecto 16:9
0x23	Subtítulos DVB (para sordos) para relación de aspecto 2.21:1
0x24	Subtítulos DVB (para sordos) para mostrar en alta definición
0x25 - 0x2F	Reservado para uso futuro
0x30	Lenguaje de signos abierto para sordos
0x31	Lenguaje de signos cerrado para sordos
0x32 - 0x3F	Reservado para uso futuro
0x40	Vídeo *up-sampled* (incremento de muestras) a partir de material con definición estándar
0x41 - 0xAF	Reservado para uso futuro
0xB0 to 0xFE	Definido por el usuario

Subtitling type	Descripción
0xFF	Reservado para uso futuro

Apéndice D

Run-length coding - RLC

La compresión RLC o Run-length coding es una forma muy simple de compresión de datos en la que, secuencias de datos consecutivos con el mismo valor, son almacenadas como un único valor, seguido del número de veces consecutivas que aparece dicho valor. Esto resulta especialmente útil en datos que contienen muchas de estas secuencias, como es el caso de los subtítulos, los iconos y los logotipos. De ahí que sea usado para el subtitulado DVB.

Existen muchas formas de aplicar esta compresión, pero, en esencia, todas se basan en un dato, seguido del contador de veces que aparece. Si trasladamos el algoritmo al escenario de subtitulado DVB, el dato pasa a ser el número de la entrada de la CLUT con el color del píxel que se desea. Atendiendo a la Figura D.1, es posible apreciar cuál sería el resultado para un caso sencillo, en el que se pretende codificar mediante este algoritmo de compresión, una imagen de dimensiones 1x17 píxeles y compuesta únicamente de 4 colores.

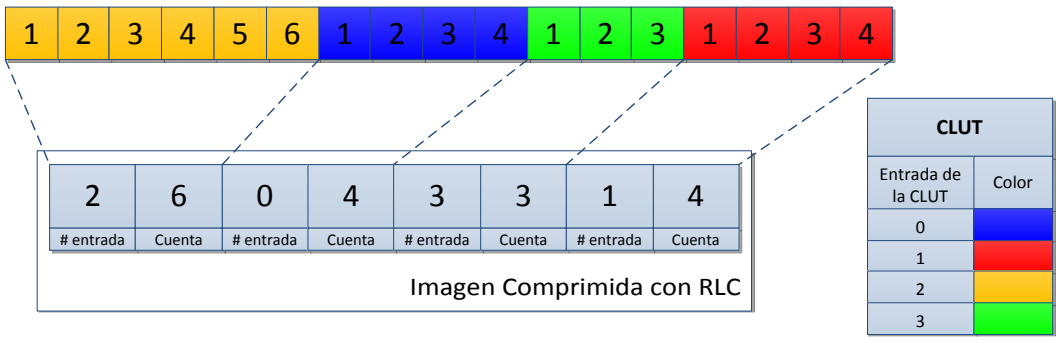


Figura D.1: Run-length Coding.

Como se puede ver, la imagen queda descrita con muy pocos bits lo que pone de manifiesto la utilidad del RLC. No obstante, para escenarios diferentes a los propuestos (Subtítulos, logotipos, iconos, etc.) este mecanismo de compresión puede resultar altamente ineficiente.

Apéndice E

Secuencias posibles de bits de un 4-bit/pixel_code_string

Tabla E.1: Secuencias posibles de bits de un 4-bit/pixel_code_string.

Secuencias posibles de un 4-bit/pixel_code_string	# Bits	Necesita alinea-ción	Descripción
CCCC	4	Sí	**1 pixel del pseudo-color de la entrada cuyo número es CCCC.**
0000 | 0 | 000	8	No	Advierte que éste es el **último 4-bit/pixel_code_string** antes del pixel-data_sub-block() cuyo data_type vale "0xF0", es decir, fin de línea de objeto.
0000 | 0 | RRR	8	No	**2 + RRR píxeles del pseudo-color de la entrada 0**. Puesto que RRR siempre va a ser mayor o igual que 1 (de lo contrario sería el mismo caso que el de la fila anterior), el rango de píxeles consecutivos que se puede cubrir está entre 3 y 9.
0000 | 1 | 0 | RR | CCCC	12	Sí	**4 + RR píxeles del pseudo-color de la entrada cuyo número es CCCC**. El rango de píxeles consecutivos va a estar entre 4 y 7.

Secuencias posibles de un 4-bit/pixel_code_string	# Bits	Necesita alinea-ción	Descripción
0000 | 1 | 1 | 0 0 |	8	No	**1 pixel del pseudo-color de la entrada 0**.
0000 | 1 | 1 | 0 1 |	8	No	**2 píxeles del pseudo-color de la entrada 0**.
0000 | 1 | 1 | 1 0 | RRRR | CCCC	16	No	**9 + RRRR píxeles del pseudo-color de la entrada cuyo número es CCCC**. El rango de píxeles consecutivos va a estar entre 9 y 24.
0000 | 1 | 1 | 1 1 | RRRRRRRR | CCCC	20	Sí	**25 + RRRRRRRR píxeles del pseudo-color de la entrada cuyo número es CCCC**. El rango de píxeles consecutivos va a estar entre 25 y 280.

Apéndice F

Formato básico BMP

Tabla F.1: Formato Básico de un archivo BMP

Nombre	Tamaño	Descripción
Cabecera Archivo	14 bytes	Estructura BITMAPFILEHEADER
bfType	2 bytes	Definición del tipo de archivo. Debe ser 'BM'.
bfSize	4 bytes	Tamaño de archivo en bytes.
bfReserved1	2 bytes	Sin usar. Fijado a 0.
bfReserved2	2 bytes	Sin usar. Fijado a 0.
bfOffBits	4 bytes	Offset hasta el Pixel data.
Cabecera Imagen	40 bytes	Estructura BITMAPINFOHEADER
biSize	4 bytes	Tamaño de la cabecera. Al menos 40 bytes.
biWidth	4 bytes	Ancho de la imagen.
biHeight	4 bytes	Alto de la imagen. Puede ser negativo.
biPlanes	2 bytes	Número de planos. Fijado a 1.
biBitCount	2 bytes	Resolución de color del DIB. 1, 4, 8, 16, 24 o 32 bits/pixel.
biCompression	4 bytes	Tipo de Compresión: 0 = BI_RGB 1 = BI_RLE8 2 = BI_RLE4
biSizeImage	4 bytes	Tamaño del bitmap en bytes. Puede ser 0.
biXPelsPerMeter	4 bytes	Resolución Horizontal: Píxeles/metro
biYPelsPerMeter	4 bytes	Resolución Vertical: Píxeles/metro
biClrUsed	4 bytes	Número de colores usados. Si es 0 se obtiene como: $2^{biBitCount}$.
biClrImportant	4 bytes	Número de colores importantes

Nombre	Tamaño	Descripción
Tabla de colores	4 * nº colores bytes	Presente si Resolución <= 8. Se repite tantas veces como colores se especifiquen
Azul	1 byte	Valor del color azul
Verde	1 byte	Valor del color verde
Rojo	1 byte	Valor del color rojo
Reservado	1 byte	Reservado. Fijado a 0.
Pixel Data	biSizeImage bytes	La información de los píxeles