TESIS DOCTORAL

2014

DISEÑO Y EVALUACIÓN DE SISTEMAS DE ESTIMACIÓN DE ANCHO DE BANDA DISPONIBLE PARA SERVICIOS ADAPTATIVOS DE VÍDEO STREAMING

LAURA POZUECO ÁLVAREZ

INGENIERA DE TELECOMUNICACIÓN

PROGRAMA DE DOCTORADO EN TECNOLOGÍAS INDUSTRIALES

GABRIEL DÍAZ ORUETA

XABIEL GARCÍA PAÑEDA

Resumen

La transmisión de contenidos multimedia a través de la red es una de las aplicaciones más demandadas actualmente. No en vano, más de la mitad del tráfico en Internet se corresponde con servicios de *streaming* de vídeo en lo que calificamos como "horas punta". Garantizar la calidad en la percepción de los contenidos frente a la gran demanda del servicio es el gran desafío al que se enfrentan los servicios de vídeo *streaming*. Además, hoy en día, el acceso a contenidos multimedia puede realizarse a través de diversos tipos de redes y con una amplia variedad de terminales que presentan diferentes restricciones. Por tanto, la adaptación de los contenidos a un entorno heterogéneo como es Internet será un proceso clave en la mejora de la percepción de la calidad del usuario.

La presente tesis doctoral aborda esta problemática, aportando soluciones para tres formas diferentes de llevar a cabo la adaptación de los contenidos. Primeramente, se presenta un estimador no intrusivo para la tecnología *streaming*, basando la adaptación de los contenidos en la transcodificación de los mismos en tiempo real. Analizando los paquetes de vídeo recibido en la aplicación del cliente, el estimador es capaz de seleccionar el *bitrate* de codificación más adecuado para el ancho de banda disponible *end-to-end*. La estimación del ancho de banda disponible se lleva a cabo con métodos no intrusivos, basados en métricas clásicas como los paquetes perdidos o el *jitter* pero también en métricas novedosas como la linealidad de los instantes de recepción de los paquetes RTP. El estimador ha sido integrado en una arquitectura real cliente/servidor y evaluado con diferentes situaciones de tráfico en la red.

La segunda solución para un sistema *streaming* adaptativo está basada en el empleo de la tecnología *Scalable Video Coding* (SVC), recientemente estandarizada como extensión del códec H.264/AVC. Empleando información de *feedback* de los clientes acerca del estado de la transmisión, el servidor es capaz de seleccionar la combinación más adecuada de capas SVC para el ancho de banda disponible. Las métricas de estimación de ancho de banda analizadas en el sistema anterior se adaptan a la tecnología SVC para perfeccionar el algoritmo de estimación. El sistema se implementa en equipos reales y los resultados muestran la correcta operación y la precisión del sistema ante diferentes variaciones del ancho de banda disponible, así como la mejora en la escalabilidad del sistema cuando al servicio acceden clientes de manera simultánea.

Por último, no podemos obviar el incremento de popularidad de las soluciones *streaming* basadas en HTTP. A pesar de que los protocolos basados en TCP no fueron considerados, en un principio, aptos para la transmisión de contenido multimedia en tiempo real, el reciente desarrollo del estándar *Dynamic Adaptive Streaming over HTTP* (DASH) proporciona una solución. De esta forma, basándose en protocolos ampliamente utilizados

en Internet, se facilita el reaprovechamiento de la infraestructura de red desplegada. En este sentido, en esta tesis se propone un último sistema *streaming* adaptativo empleando dicha tecnología. Al igual que en las propuestas previas, el sistema se implementará en equipos reales y se analizarán las mejoras introducidas.

Palabras clave: *streaming*, vídeo de alta calidad, estimación de ancho de banda, adaptación, transcodificación, *Scalable Video Coding* (SVC), *Dynamic Adaptive Streaming over HTTP* (DASH).

Abstract

The transmission of multimedia content over the network is one of the most demanded applications. Not surprisingly, more than half of Internet traffic corresponds to video streaming services in the "peak hours". Guarantee the quality of the contents in spite of the great demand of the service is the main challenge that video streaming services face. Moreover, access to multimedia content can be done through various kinds of networks and a wide variety of terminals with different constraints. Therefore, the adaptation of the contents to a heterogeneous environment such as the Internet is a key process in improving the perceived quality of the user.

This thesis addresses this issue, providing solutions for three different ways to carry out the adaptation process. First, a non intrusive estimator, based on transcoding techniques, for streaming technology is presented. Analyzing the video packets received by the client application, the estimator is capable of selecting the most suitable encoding bitrate for the available bandwidth in the end-to-end path. Estimation of the available bandwidth is performed with non-intrusive methods based on conventional metrics such as jitter or packet loss but also in novel metric as the linearity of the instants of reception of RTP packets. The estimator has been integrated in a real client / server architecture and evaluated with different network traffic situations.

The second solution to an adaptive streaming system is based on the use of Scalable Video Coding (SVC) technology, recently standardized as an extension of H.264 / AVC. Using feedback information from clients about the transmission status, the server is able to select the most suitable combination of SVC layers for the available bandwidth. Bandwidth estimation metrics are adapted to the SVC technology, improving the estimation algorithm. The system is implemented in real equipment and the results show the correct operation and accuracy of the system when adapting to different variations of the available bandwidth.

Also, the system scalability is improved when clients access the service simultaneously.

Finally, we can not ignore the increasing popularity of HTTP-based streaming solutions. Although protocols based on TCP were not considered suitable for the transmission of multimedia content in real time, the recent development of the standard Dynamic Adaptive Streaming over HTTP (DASH) provides a solution. Thus, based on widely used protocols on the Internet, the reuse of deployed network infrastructure is provided. Here, in this thesis a final adaptive streaming system is proposed using such technology. As in previous proposals, the system is implemented in real equipment and the improvements will be discussed.

Keywords: *streaming*, high quality video, bandwidth estimation, adaptation, transcoding, *Scalable Video Coding* (SVC), *Dynamic Adaptive Streaming over HTTP* (DASH).

Agradecimientos

A mis directores de tesis, Gabriel y Xabi, por su constante dedicación y consejos.

A todo el equipo de investigación, con el que compartí conocimiento y experiencias.

A mi familia, por su apoyo incondicional.

Índice general

Índice de figuras

Índice de Tablas

Parte I

Introducción y objetivos

Capítulo 1

INTRODUCCIÓN

Los últimos años han supuesto un gran cambio en cuanto a la forma de consumo de material audiovisual y cada vez es más frecuente que el usuario elija el momento y el lugar para visualizar los contenidos. Este fenómeno supone una migración de la televisión tradicional hacia el consumo multimedia en Internet.

En los primeros sistemas de transmisión de audio o vídeo en Internet era necesario que los contenidos se enviasen por completo antes de poder reproducirlos. Por tanto, un requisito indispensable era disponer de suficiente espacio de almacenamiento y esperar a que finalizase la descarga para poder ver/oír el contenido del archivo. Como solución a estos problemas se planteó la distribución de este tipo de información en tiempo real, surgiendo así la tecnología *streaming*.

Con la tecnología *streaming* (bien sea en su modalidad en directo o bajo demanda) podemos transmitir información multimedia de manera continua, reproducirla en tiempo real y posteriormente descartarla. De esta manera se reduce el tiempo de espera necesario para comenzar el visionado, puesto que la reproducción se puede realizar mientras el archivo se está descargando y además se ahorra espacio en disco. Todas las ventajas que presenta esta nueva forma de consumo de contenidos multimedia son posibles gracias al desarrollo de nuevos protocolos pensados para la transmisión de información en tiempo real. La Figura 1.1 resume el comportamiento de la transferencia clásica de archivos y la transferencia mediante *streaming*, apreciando las características y diferencias destacadas en párrafos anteriores de manera gráfica.

Sin embargo, la tecnología *streaming* presenta debilidades relacionadas con la calidad percibida en el lado del cliente puesto que el servicio depende en gran medida del estado de la red. Si los enlaces de red sufren descensos de velocidad, se podría continuar la reproducción con la información almacenada en el *buffer* de precarga del cliente. Sin embargo, si los problemas persisten, el *buffer* del cliente puede llegar a vaciarse, produciendo así cortes durante la reproducción. En consecuencia, es necesario contar con unas condiciones estables para la reproducción.

Por otro lado, las redes IP (*Internet Protocol*) utilizan la conmutación de paquetes. En lugar de dedicar un enlace exclusivo para cada comunicación, el origen envía los mensajes en paquetes. Los paquetes son el resultado de fragmentar la información que se desea

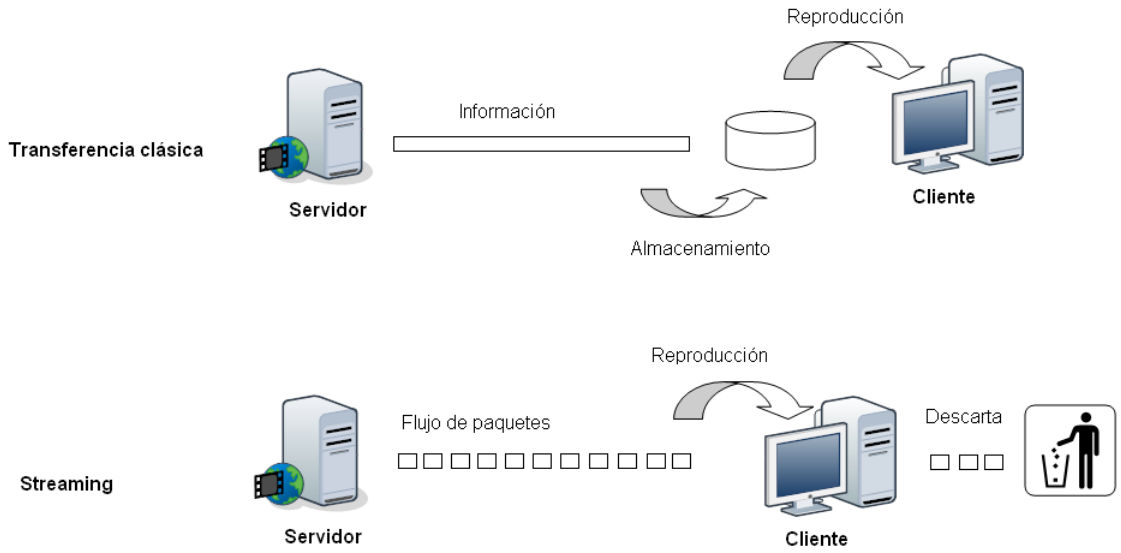


Figura 1.1: Transferencia clásica vs. *streaming*

transmitir y a cada uno de ellos se le añade una cabecera con información suficiente para ser enrutado al destino correspondiente. Cada paquete se trata de manera independiente, de forma que paquetes que pertenecen al mismo mensaje pueden seguir caminos diferentes en función del nivel de congestión que haya en la red. Una vez en el destino, los paquetes son reensamblados para recomponer la información original. Este tipo de redes presentan el inconveniente de que no hay garantía de calidad de servicio, ya que no hay una reserva de recursos antes de establecer la comunicación.

Todo esto tiene especial relevancia en las líneas de acceso del usuario debido, principalmente, a las limitaciones que existen en la tasa binaria (Figura 1.2). Generalmente, las líneas de acceso presentan un alto porcentaje de ocupación, así que las prestaciones máximas dependen de los servicios y del número de usuarios compitiendo por los recursos. Factores como la pérdida de paquetes, los retardos y la congestión de red afectan de manera directa a la calidad de presentación del vídeo. Por tanto, el gran desafío en la transmisión de contenido multimedia en Internet es la adaptación de la calidad de los contenidos con el objetivo de mejorar la experiencia de usuario. Como hemos comentado, el servicio *best-effort* que está disponible actualmente en Internet no proporciona garantías, expresadas en términos de mínimo ancho de banda o máximo retardo o pérdidas, asociadas con las aplicaciones en tiempo real. Sin embargo, una manera de que este tipo de aplicaciones sean soportadas en redes *best-effort* es usar mecanismos adicionales de control que adapten los procesos de codificación, transmisión, recepción o decodificación dependiendo del estado de la red. La idea es simple: maximizar la calidad en el lado del receptor.

Algunas plataformas comerciales incorporan mejoras para paliar estos efectos de degradación de la calidad bajo ciertas condiciones. Por ejemplo, *SureStream*, de *Real Networks*[1] se consolidó como una de las primeras soluciones comerciales en implementar cierto grado

[1] http://www.realnetworks.com/

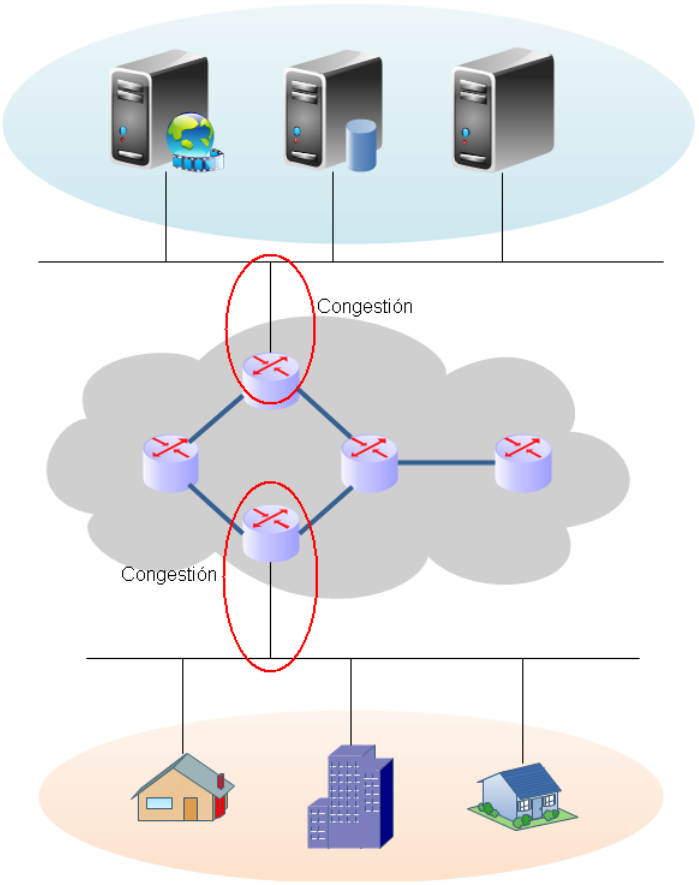


Figura 1.2: Limitaciones en las líneas de acceso

de adaptación en la transmisión de los contenidos. Por su parte, *Adobe*[2], *Microsoft*[3] y *Apple*[4], o empresas como *Vidyo*[5] entre otros, también se han embarcado en la comercialización de soluciones *streaming* adaptativas. Sin embargo, la transmisión *streaming* a través de Internet todavía se encuentra en etapas iniciales con respecto a lo que podría llegar a ser. La principal razón de todo ello es la limitación que imponen estos productos comerciales debido a la ausencia de interoperabilidad entre las diferentes propuestas y la necesidad de disponer de las herramientas propietarias para acceder al servicio. Por esta razón, consideramos necesario buscar soluciones basadas en estándares abiertos, que permitan así el crecimiento y expansión del mercado a partir de un ecosistema común que permita el uso de diferentes dispositivos para acceder al servicio.

Esta tesis doctoral aborda el diseño, implementación y análisis de servicios *streaming* adaptativos, estudiando diferentes opciones y tecnologías que permitan ajustar la calidad de los contenidos transmitidos al ancho de banda disponible. En concreto analizaremos tres maneras de llevar a cabo la adaptación. Por un lado, estudiaremos una primera aproximación basada en técnicas de transcodificación, mediante las cuales el servidor modifica en tiempo real el *bitrate* de codificación de los contenidos que se envían al cliente.

[2]http://www.adobe.com/es/products/hds-dynamic-streaming.html
[3]http://www.iis.net/downloads/microsoft/smooth-streaming
[4]http://tools.ietf.org/html/draft-pantos-http-live-streaming-13
[5]http://www.vidyo.com/

En segundo lugar analizaremos el empleo de codificadores escalables tales como SVC (*Scalable Video Coding*), que permite codificar un fichero de vídeo en diferentes calidades dentro del mismo *bitstream*. Esto incluye diferentes resoluciones, diferentes frecuencias (fps) y diferentes calidades, (bits por píxel), permitiendo así la adaptación del vídeo transmitido al ancho de banda disponible cambiando la resolución espacial, temporal y/o de calidad. Por último, incluiremos en el estudio el estándar DASH (*Dynamic Adaptive Streaming over HTTP*) surgido para la transmisión multimedia empleando el protocolo HTTP (*Hypertext Transfer Protocol*), considerado en sus inicios un protocolo no viable para la transmisión de contenido multimedia pero que, gracias a las mejoras aportadas por el estándar DASH, se está posicionando como un claro rival al *streaming* tradicional.

Capítulo 2

MOTIVACIÓN

Desde sus orígenes, Internet ha experimentado una constante evolución tecnológica, estrechamente asociada al crecimiento de usuarios y al aumento del ancho de banda. En relación al último punto, la velocidad media de las conexiones en Internet se situaba en torno a 8 Mbps en el año 2011, aumentando esta cifra a 11 Mbps en el siguiente año. Siguiendo con este crecimiento exponencial, se estima que en el año 2018, el ancho de banda medio pase a situarse en torno a los 42 Mbps. De la mano de este aumento de ancho de banda se sitúa el aumento de usuarios conectados: en 2012 cerca del 32 % de la población mundial disponía de acceso a Internet. Las predicciones para el año 2018 sitúan este porcentaje próximo al 51 % [1].

Fruto de la evolución tecnológica que experimenta constantemente Internet, es la aparición de nuevas tecnologías que han permitido agilizar el proceso de intercambio de datos. Por ejemplo, gracias a protocolos como UDP (*User Datagram Protocol*) o RTSP (*Real Time Streaming Protocol*) y gracias también al aumento del ancho de banda en las líneas de acceso del usuario, se han podido incorporar los formatos de vídeo en los procesos de transmisión de datos, dando lugar a la tecnología *streaming*. Más recientemente, la aparición de nuevos estándares permiten el empleo de protocolos de transporte que tradicionalmente no se consideraban aptos para el envío de contenido multimedia en tiempo real, como el protocolo HTTP (*Hypertext Transfer Protocol*).

Desde la introducción de los primeros productos comerciales en los años 90, la transmisión de contenido multimedia a través de la web se ha posicionado como uno de los servicios más influyentes hoy en día. Uno de los motivos del éxito de *streaming* radica en el poco tiempo de espera y en el ahorro de espacio, ya que podemos ver y escuchar el archivo mientras se descarga. Muchas cadenas de televisión y radio emplean esta tecnología para difundir sus contenidos a través de Internet y transmitir eventos en directo, e incluso los propios usuarios pueden difundir sus vídeos. Por su parte, *Cisco Systems* (Cisco, 2014) prevé una tasa de crecimiento anual (*Compound annual growth rate*, CAGR) del 69 % para aplicaciones de vídeo en dispositivos móviles en el periodo comprendido entre los años 2013-2018. Estamos, por tanto, ante un mercado que ofrece grandes expectativas y se espera que el vídeo *streaming* sea la *«killer-application»* en las redes de próxima generación, convirtiendo al contenido de vídeo en el tráfico predominante en Internet, con una

[1] http://www.cisco.com/c/en/us/solutions/service-provider/visual-networking-index-vni/index.html#~overview

cuota del 79 % del tráfico total en el 2018.

Pero la transmisión vía *streaming* de contenidos no está exenta de problemas. El incremento de la popularidad de los servicios de audio y vídeo en Internet viene acompañado de la exigencia de mayor calidad de imagen por parte de los usuarios: los usuarios cada vez hacen un uso más intensivo de este tipo de servicios y los proveedores intentan ofrecer vídeo de mejor calidad para responder a esas necesidades. Sin embargo, la principal limitación de esta tecnología se encuentra en la necesidad de disponer de unas condiciones de transmisión estables para poder garantizar una cierta calidad de servicio (QoS, *Quality of Service*): el ancho de banda es uno de los factores clave para la tecnología *streaming*.

Desgraciadamente, mantener las condiciones óptimas en la calidad percibida no resulta trivial para servicios con requisitos en tiempo real en redes como Internet, ya que no se tiene control sobre el enrutamiento de los datos y pueden darse situaciones de elevada latencia y pérdida de paquetes durante la transmisión. Las redes de datos como Internet están basadas en el principio *best-effort* y, por tanto, no ofrecen ningún tipo de QoS. El tráfico de vídeo, por su parte, tiene que cumplir con una serie de requerimientos que no son necesarios en el tráfico de datos. En particular, el vídeo *streaming* tiene unos requerimientos estrictos en cuanto a latencia extremo a extremo y variación del retardo. Un retardo extremo a extremo elevado afectará al envío de los datos de vídeo, provocando cortes durante la reproducción con la consecuente frustración del usuario final. De ahí que el ancho de banda disponible y su estabilidad puedan convertirse en el factor limitante de esta tecnología.

El punto más conflictivo lo encontramos en las líneas de acceso del usuario, donde las condiciones de transmisión varían, afectando a la calidad de las reproducciones. Además, el porcentaje de ocupación en las redes de acceso hace que la capacidad necesaria para lograr una transmisión fluida no sea suficiente en determinados momentos. Surge así la necesidad de adaptar los contenidos de vídeo a las diferentes capacidades de las distintas redes de acceso.

No es de extrañar que estos desafíos, en conjunción con las promesas comerciales de la tecnología, hayan atraído esfuerzos considerables en la investigación, particularmente dirigidos a la eficiencia en la transmisión (Civanlar et. al, 2001).

Con el objetivo de solucionar los problemas derivados del aumento de la resolución de los vídeos y del incremento de usuarios, se ha trabajado en los últimos años en diversos elementos que podrían hacer que los servicios de Internet TV o Internet VoD (*Video on Demand*) sean más flexibles ante problemas puntuales de capacidad en las redes de comunicaciones. Por consiguiente, para proporcionar un servicio de vídeo *streaming* de calidad, se tienen que tener en cuenta otros componentes adicionales a la arquitectura clásica, como podrían ser un módulo de estimación del ancho de banda y un controlador de tasa adaptativo.

Gracias al empleo de mecanismos adicionales, podremos ajustar los contenidos multimedia transmitidos al ancho de banda disponible. La transcodificación es una de las técnicas que nos permiten llevar a cabo esta tarea, variando la tasa de transmisión de los

contenidos en tiempo real y de acuerdo a la capacidad de la red. Además, los parámetros como la tasa de transmisión, los *frames* por segundo o la resolución espacial también se pueden modificar con técnicas de codificación escalable como SVC (*Scalable Video Coding*) o con mecanismos de *streaming* adaptativo sobre HTTP como DASH (*Dynamic Adaptive Streaming over HTTP*).

En este sentido, en esta tesis doctoral se proponen diferentes soluciones *streaming* adaptativas para diferentes tecnologías. Todas las soluciones propuestas incluyen mecanismos de estimación del ancho de banda disponible que permiten ajustar la calidad de los contenidos transmitidos. Mediante la inclusión de los sistemas de estimación de la congestión y adaptación de los contenidos, se evitan situaciones en las que la calidad percibida por el usuario se ve perjudicada a consecuencia de pérdidas y retardos sufridos por efecto de la congestión en la red.

Capítulo 3

OBJETIVOS

Uno de los principales problemas en la construcción de sistemas *streaming* adaptativos es la correcta estimación del ancho de banda disponible. Para solventar los desafíos que plantea el cálculo de los recursos libres en la red en cada instante, es necesario analizar el proceso de transmisión del vídeo para estudiar el comportamiento del tráfico en diversas situaciones de congestión. De esta manera se podrán extraer los parámetros que están directamente relacionados con la disponibilidad de recursos en la red. La dependencia que se establezca entre los parámetros analizados y la congestión en la red nos permitirá construir algoritmos que permitan estimar el ancho de banda disponible en cada momento, para el posterior ajuste de la calidad de los contenidos transmitidos.

Para el desarrollo de los diferentes algoritmos de estimación de ancho de banda será necesario evaluar diferentes parámetros de congestión. Además, el análisis del comportamiento del tráfico *streaming* deberá realizarse en una maqueta real y bajo diferentes patrones de variación del tráfico en la red. Los parámetros, observados de forma no intrusiva como consecuencia de la transmisión del propio contenido multimedia, se emplean para realimentar el algoritmo diseñado con el objetivo de estimar los recursos disponibles para cada usuario. A partir del resultado fruto de la estimación, se podrá adaptar la calidad del flujo transmitido al ancho de banda disponible, con el fin de evitar la pérdida de paquetes y una mayor degradación de la calidad de la experiencia (QoE) del usuario.

La adaptación de los contenidos se realizará de manera diferente en función de la tecnología *streaming* bajo estudio. Por ejemplo, para un sistema basado en transcodificación deberemos modificar la características del *bitrate* de codificación. Para sistemas streaming basados en codificadores escalables, la adaptación se realiza eliminando o añadiendo capas que conforman el *bitstream* de vídeo, atendiendo a la filosofía de SVC (*Scalable Video Coding*). En el caso de sistemas basados en HTTP *streaming*, como DASH, la adaptación se realiza modificando la representación a la que pertenece el segmento que solicita el cliente.

El estudio y las mejoras propuestas para los sistemas de vídeo adaptativo tienen una aplicación práctica muy importante en sectores relacionados con la distribución de contenidos multimedia de alta calidad, ya que permite dotar de cierta QoS a los servicios de vídeo *streaming*, adaptando la tasa de transferencia a la disponibilidad de ancho de banda en la red de acceso del usuario, donde las condiciones del canal son cambiantes y los recursos son limitados.

Según todo lo anterior, los objetivos globales de este trabajo serán el análisis de la tecnología *streaming*, el estudio de las características y el comportamiento de la transmisión de contenido multimedia en entornos de red con condiciones variables y, por último, el diseño, implementación y evaluación de diferentes soluciones de adaptación de los contenidos al ancho de banda disponible.

De forma detallada, los objetivos que se persiguen son los siguientes:

1. Estudiar el estado del arte y la tecnología *streaming*. Un aspecto de vital importancia es el conocimiento de los protocolos y los aspectos técnicos de la tecnología *streaming*, así como el análisis de los diferentes trabajos relacionados que existen en el campo de la investigación.

2. Establecer y definir los parámetros clave que caractericen los servicios de vídeo. Estos indicadores podrán ser el retardo, la variación del retardo y el porcentaje de paquetes perdidos, entre otros. El estudio de estos indicadores es crucial para poder determinar la calidad de la comunicación.

3. Diseñar los entornos de prueba. Con los entornos de prueba recrearemos diferentes situaciones que permitan evaluar cada una de las tecnologías y estudiar su comportamiento sobre una maqueta real.

4. Diseñar algoritmos de estimación y adaptación para las diferentes tecnologías *streaming* estudiadas. En el diseño del algoritmo de estimación y adaptación tendremos que tener en cuenta factores como la sobrecarga introducida en la red, o la propia sobrecarga introducida en los equipos a costa de la implementación de los módulos adicionales de estimación y adaptación. Un aspecto de vital importancia para cualquier sistema de estas características es la correcta estimación del ancho de banda disponible, de manera que la adaptación de los contenidos no generen situaciones de inestabilidad en el empleo del ancho de banda.

5. Integrar los algoritmos diseñados en arquitecturas reales de distribución cliente/servidor para cada una de las tecnologías *streaming* estudiadas. Dichas implementaciones se realizarán a partir de estándares abiertos y proyectos de código libre para servidores y clientes *streaming*, permitiendo así la integración en un entorno heterogéneo.

6. Evaluar las propuestas. Los sistemas adaptativos resultantes serán evaluados empleando *streams* de vídeo reales en escenarios emulados, lo que permite un excelente control de las condiciones de transmisión.

Capítulo 4

ESTRUCTURA DE LA MEMORIA

El presente documento se divide en 5 partes claramente diferenciadas. Cada una de las partes se corresponde con un aspecto del estudio de investigación diferente.

- Parte I: *Introducción y objetivos*. Engloba los capítulos iniciales referentes a la introducción al tema de la tesis en el «Capítulo 1 Introducción», así como la motivación que nos lleva a plantear el presente trabajo de investigación en el «Capítulo 2 Motivación». Los objetivos se enumeran en el «Capítulo 3 Objetivos».

- Parte II: *Estudio de la tecnología*. En esta sección se describe brevemente toda aquella información teórica que pueda resultar útil para el lector a la hora de analizar e interpretar el trabajo desarrollado en la parte IV de la memoria. La división en capítulos de este apartado viene marcado por las áreas temáticas que cubre. Por un lado, el «Capítulo 5 Vídeo Digital» resume las principales nociones acerca del vídeo en formato digital. La tecnología *streaming* como tal es esbozada en el «Capítulo 6 Tecnología *Streaming*». Y, por último, las principales características de las aplicaciones multimedia en la red se analizan en el «Capítulo 7 Aplicaciones Multimedia en la Red».

- Parte III: *Trabajos relacionados*. En este apartado se hace un recorrido a lo largo de los principales trabajos de otros autores que presentan ciertos puntos en común con los objetivos marcados en esta tesis doctoral.

- Parte IV: *Aportaciones*. Aglutina el trabajo de investigación realizado. Está divido en cuatro capítulos de los cuales, el primero de ellos, correspondiente al «Capítulo 9 Consideraciones Previas», introduce la metodología de trabajo. Los tres capítulos restantes incluyen el diseño, desarrollo, implementación y análisis para cada una de las opciones adaptativas planteadas. El «Capítulo 10 Estimación y Adaptación Empleando Técnicas de Transcodificación» detalla el sistema adaptativo basado en transcodificación. La continuación y mejora de este primer estimador se propone en el «Capítulo 11 Estimación y Adaptación Empleando Codificadores Escalables», con un sistema adaptativo basado en codificadores escalables. Finalmente, estudiamos las opciones de transmisión de contenido multimedia mediante HTTP y, en el «Capítulo 12 Estimación y Adaptación para HTTP *Streaming*», se analiza una solución para este sistema.

- Parte V: *Conclusiones y trabajos futuros*. Para finalizar, la última parte de la memoria resume en el «Capítulo 13 Conclusiones» los principales hitos y conclusiones

detallados a lo largo de toda la documentación. Los trabajos derivados de los resultados obtenidos en este estudio se incluyen en el «Capítulo 14 Trabajos Futuros», donde también se introducen las directrices que, presumiblemente, marcarán el futuro de los servicios de vídeo.

El documento finaliza con el Capítulo 15, donde se lista y resume todas las publicaciones derivadas del desarrollo de esta tesis doctoral.

Por último, los apéndices incluyen la lista del glosario de términos empleados a lo largo de la memoria (Apéndice A) o la arquitectura de los sistemas descritos en la Parte IV (Apéndice B). También se recoge un resumen de información técnica referente a las herramientas empleadas y a la configuración de los escenarios empleados en el proceso de evaluación (Apéndice C) y análisis (Apéndice D).

Parte II

Estudio de la tecnología

Capítulo 5

VIDEO DIGITAL

El flujo continuo que capta una cámara de vídeo es de naturaleza analógica y tiene unas tasas de transferencia muy elevadas, en torno a los 166 Mbps. Las líneas de acceso presentan una capacidad muy por debajo de dicho valor, por lo que se hace necesario un proceso previo de digitalización donde se adecúan los contenidos para su transmisión a través de la red de datos mediante técnicas de codificación y compresión.

Un vídeo digital consiste en una secuencia de imágenes, cada una de ellas separada de la anterior por un factor constante. Cada una de esas imágenes es un *frame* que deberemos transmitir. Generalmente se emplean 25 imágenes por segundo para dar sensación de movimiento, sin embargo, en algunos casos, donde la secuencia de imágenes presenta un patrón principalmente estático (como puede ser una videoconferencia), podemos conseguir este efecto con 15 imágenes por segundo. Este número de imágenes por unidad de tiempo recibe el nombre de frecuencia de imagen o *frame rate*.

La calidad del vídeo resultante del proceso de codificación vendrá marcada por una serie de factores tales como el tamaño del *frame* (altura y anchura en píxeles), el número de imágenes por segundo o el número de bits que se empleen para representar el color de la imagen. Además, la tasa de bits empleada es otro factor determitante en la calidad final del vídeo digital.

En los sucesivos apartados resumimos algunos conceptos claves referentes a los procesos de codificación y la compresión de las imágenes digitales.

5.1. Códecs, técnicas de compresión

La codificación y las técnicas de compresión resultan claves en los formatos de vídeo digital. Su finalidad es digitalizar y reducir el tamaño de la información original a base de eliminar información redundante o que no resulta importante para una posterior reconstrucción. Se consigue así que las imágenes se puedan representar con una distorsión visual aceptable y un ratio de compresión elevado. Este proceso es totalmente necesario, ya que los contenidos de vídeo originales presentan un elevado volumen de información que resulta inadecuado para la transmisión por una red de datos.

Existen varias técnicas para llevar a cabo este proceso. Por ejemplo, una de ellas consis-

te en eliminar únicamente la información redundante, de manera que no existen pérdidas entre la información original y la reconstruida. En otros casos se producen pérdidas y los contenidos originales y los reconstruidos no son iguales, ya que durante el proceso de compresión se eliminó parte de la información original. De esta manera, establecemos una clasificación de los algoritmos de compresión en función de la fidelidad de la secuencia comprimida resultante, definiendo así algoritmos sin pérdidas (*lossless*) y con pérdidas (*lossy*).

En los siguientes apartados se describen brevemente las características principales de la codificación MPEG-4 (*Motion Pictures Experts Group*) Parte 10, un tipo de códec ampliamente utilizado que emplea técnicas de compresión con pérdidas y H.264/SVC, un nuevo estándar de codificación escalable.

5.1.1. Codificación MPEG-4 Part 10 (H.264/AVC)

MPEG-4 [ISO/IEC 14496] (Wiegand et. al, 2003) es el estándar de codificación más implantado en lo referente a contenidos que serán transmitidos por Internet. Su principal ventaja está en la posilibidad de distribuir contenidos desde las tasas más bajas (como en teléfonos móviles) hasta tasas elevadas (en calidad DVD). Este amplio rango de funcionamiento permite que pueda ser utilizado en varios tipos de aplicaciones, como videoconferencias, distribución de TV y, por supuesto, distribución de contenidos a través de *streaming*.

A diferencia de otros codificadores, H.264 es un estándar abierto, lo que quiere decir que, a partir de la especificación del códec, se puede construir el codificador y el decodificador. Esto también quiere decir que se pueden encontrar multitud de codificadores comerciales y libres. Sin embargo, el uso del estándar H.264 no es totalmente gratuito ya que muchas de las tecnologías que contribuyen al desarrollo del codificador están patentadas, por lo que es regulado mediante políticas de licencias.

Como comentamos anteriormente, el vídeo codificado consta de una serie de imágenes o *frames*. Mediante los algoritmos de compresión y codificación empleados en MPEG-4, se establece una clasificación de los *frames*, dando lugar a 3 tipos de *frames*:

- *Frames* tipo I (*Intra Coded Pictures*): son *frames* que se codifican de manera independiente, sin hacer referencia a otras imágenes. Su nivel de compresión no es muy elevado.
- *Frames* tipo P (*Predictive Coded Pictures*): este tipo de *frames* se codifican en base a diferencias entre los *frames* anteriores. Su nivel de compresión es más elevado que para los *frames* tipo I.
- *Frames* tipo B (*Bidirectional-predictive Codec Pictures*): en este caso la información se codifica como diferencias con respecto a *frames* anteriores y posteriores. Son los *frames* que presentan el nivel de compresión más elevado. Sin embargo, como contrapartida, este tipo de *frames* presentan unos niveles de complejidad más elevados en los procesos de codificación y decodificación.

La Tabla 5.1 presenta un resumen comparativo entre las características principales de los diferentes tipos de *frames*.

	I frame	P frame	B frame
Ratio de compresión	Bajo	Bueno	Muy bueno
Complejidad	Normal	Elevada	Muy elevada

Tabla 5.1: Tipos de *frames*

Los códecs de vídeo emplean la estructura de *frames* denominada GoP (*Group of Pictures*) que consiste en un *frame* de referencia codificado independientemente (*frame* I) seguido de una secuencia de *frames* P y B en los cuales sólo hay cambios de movimiento respecto al *frame* de referencia previo codificado. A su vez, cada *frame* se subdivide en unidades menores para lograr mayor rapidez durante el proceso de compresión. Estas unidades más pequeñas son los bloques, macrobloques, *slices* y GoB (*Group of Blocks*). La unidad mínima que permite que las operaciones de compresión sean lo más sencillas posibles son los bloques, formados por muestras de tamaño 8x8 como unidad de tratamiento de la información. Los macrobloques se corresponden con agrupaciones de bloques contiguos. Y por último, los GoB y *slices* agrupan macrobloques. Gracias a esta estructura jerarquizada se consigue reducir la complejidad del proceso de compresión en MPEG.

A modo de breve introducción y sin entrar en detalles, puesto que no es objeto de estudio en esta tesis doctoral, el proceso de codificación consiste en varias fases: en una primera fase se realiza una transformación para cada bloque del dominio espacial al de la frecuencia. Este proceso se lleva a cabo mediante la transformada discreta del coseno (DCT, *Discrete Cosine Transform*), cuyos coeficientes son reales. Posteriormente se aplica un proceso de cuantización a los coeficientes. Y por último, para lograr una mayor compresión, se aplica una codificación por entropía. Comentar que, en función de los niveles de cuantización empleados, la imagen que se obtiene tras la decodificación presentará un determinado grado de parecido con la imagen original.

En la mayoría de los esquemas de codificación, la redundancia temporal se explota aplicando predicción por compensación del movimiento y, la redundancia espacial, aplicando la transformada discreta del coseno (DCT).

Para el proceso de decodificación, además de los pasos inversos al proceso de codificación, es necesario determinar qué tipo de compresión se aplicó (temporal o espacial) y emplear técnicas predictivas en el caso de la compresión temporal.

Es necesario hacer notar que codificar los contenidos con una mayor calidad supondrá un consumo de recursos mayor, tanto computacionales como de red, a la hora de transmitirlos. Los usuarios dispondrán de una línea de acceso de una capacidad dada y el acceso a contenidos que consuman más ancho de banda supondrá serias penalizaciones en lo referente a la calidad percibida por el usuario. Por tanto, habrá que emplear los parámetros de codificación adecuados para que se llegue a un compromiso entre calidad del vídeo y consumo de recursos.

5.1.2. H.264/SVC - Scalable Video Coding

El concepto de escalabilidad en la codificación de vídeo ha sido un tema ampliamente discutido en la literatura en pasados años. A medida que la tecnología avanzaba y con la reencarnación de la idea de proporcionar varios niveles de calidad en un único stream de vídeo y con los nuevos avances en codificación, surge el estándar SVC (*Scalable Video Coding*) (Schwarz et. al, 2007), que permite codificar un fichero de vídeo en diferentes calidades dentro del mismo *bitstream*. Esto incluye diferentes resoluciones (tamaño del *frame*), diferentes frecuencias (*fps, frames per second*) y diferentes tasas de bits. Es decir, SVC permite escalabilidad espacial, temporal y de calidad respectivamente. Con esto se consigue que la adaptación se realice tanto para los requerimientos de la aplicación (*display* o capacidades de procesamiento de los dispositivos) como para las variaciones en las condiciones de transmisión.

H.264/SVC se estandariza como una extensión al conocido códec de vídeo H.264/AVC (MPEG-4 Parte 10), un estándar de compresión de vídeo ampliamente utilizado en Internet o en plataformas de vídeo *streaming*. La principal limitación de H.264/AVC es que no es lo suficientemente escalable para muchos servicios, ya que, por ejemplo, la transmisión de vídeo a través de Internet está expuesta a condiciones de transmisión variables, que podrían ser tratadas usando prestaciones de escalabilidad. Por ejemplo, el actual estándar H.264/AVC es sensible a las pérdidas de paquetes que provocan errores en la transmisión. La opción para reducir este tipo de errores pasa por incrementar los anchos de banda, solución costosa y poco eficiente, sobre todo si tenemos en cuenta que este tipo de errores es debido a la congestión del tráfico.

Además, el códec H.264/AVC es un códec de una sola capa, es decir, un *stream* de vídeo codificado con AVC permanecerá con el mismo caudal durante todo el proceso de comunicación ya que parámetros tales como el *bit rate*, *frame rate*, tamaño de la imagen, etc. se eligen al inicio de la sesión. Sin embargo, los emisores y receptores pueden tener diferentes capacidades y las condiciones de la red (ancho de banda, retardo o *jitter* -variaciones del retardo-) pueden variar a lo largo de la sesión. Por tanto, si queremos soportar dispositivos finales heterogéneos, sería necesario disponer de diferentes ficheros de vídeo codificados con distintas calidades o recurrir a técnicas de transcodificación.

El estándar SVC extiende el estándar AVC con las prestaciones de escalabilidad a nivel de *stream* antes mencionadas (temporal, espacial y de calidad), de manera que la adaptación del *bitstream* a las condiciones de red y de los terminales es relativamente sencilla. La estandarización de SVC se finalizó en el año 2007 y el grupo de estandarización proporciona software de referencia (JSVM[1]) para codificar y decodificar.

Las múltiples capas que puede presentar un vídeo codificado con SVC comprenden una capa base y varias capas de mejora. La capa base se codifica con una calidad base que garantice que incluso el receptor con menos capacidades lo pueda reproducir. Además, dicha capa base es compatible con el perfil no escalable H.264/AVC. Las capas de mejora, por su parte, permiten incrementar la resolución temporal (es decir, el *frame rate*), la resolución espacial o la calidad de los contenidos del vídeo.

[1] http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm.

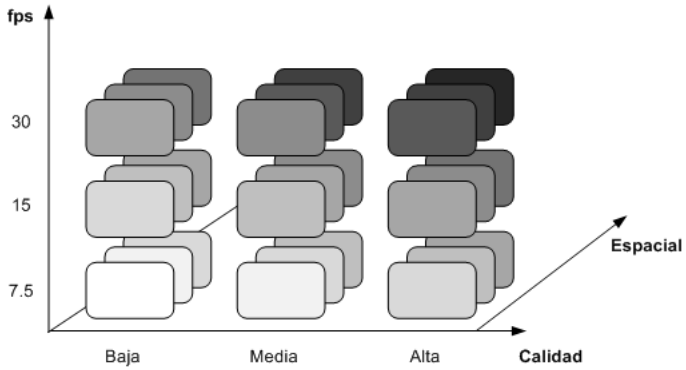


Figura 5.1: Niveles de escalabilidad en SVC

Cuando las condiciones así lo imponen, las capas superiores se descartan a lo largo de los enlaces o en elementos intermedios, mediante operaciones de baja complejidad y sin necesidad de recurrir a técnicas de transcodificación. Si se reciben todas las capas de mejora entonces el *bitstream* alcanza la calidad original de codificación.

La idea de codificación en múltiples capas que aplica SVC no es una idea totalmente nueva, ya que el concepto de escalabilidad ya fue presentado en otros estándares en la forma de perfiles escalables (H.262/MPEG2, H.263 y MPEG4). Las propuestas anteriores de escalabilidad espacial y de calidad de esos estándares presentaban el problema del incremento de la complejidad computacional del decoder y la reducción de la eficiencia de codificación si lo comparamos con el correspondiente perfil no escalable, de ahí que los perfiles escalables de estos estándares rara vez se utilizasen. Con SVC se consigue la escalabilidad a un coste computacional abordable ya que este sistema permite una adaptación eficiente directamente a nivel de *bitstream*, sin la necesidad de transcodificar o recodificar, pudiendo extraer las capas con operaciones de baja complejidad. Dichas operaciones consisten en eliminar partes del *bitstream* escalable, dando como resultado un sub-stream que forma otro *bitstream* válido, que presenta un tamaño de imagen menor, frame rate o calidad menor en comparación con el *bitstream* original.

En la Figura 5.1 podemos ver la idea que subyace en la filosofía SVC y que ha sido planteada en párrafos anteriores: dentro del mismo *stream* de vídeo se pueden añadir o eliminar capas correspondientes a diferentes niveles de frame rate o resolución espacial (CIF, QCIF ...) o calidad. La capa base, correspondiente a los menores niveles, presenta el menor consumo de ancho de banda. A medida que se añaden capas, se mejora la calidad de los contenidos transmitidos a la vez que se incrementa el consumo de ancho de banda.

Las ventajas de SVC pueden verse desde dos puntos de vista: por un lado, en términos de optimización de ancho de banda y reducción en el tamaño de almacenamiento en los servidores, además de la disminución computacional en el procesado de vídeo dentro de las redes y la adaptación en vivo en función de las variaciones en las condiciones de transmisión. Y por otro lado, desde el punto de vista del cliente, adaptando los servicios multimedia a los dispositivos y a las diferentes capacidades del terminal.

"Codifica una vez, sirve muchas" es la frase que resume SVC. Consideremos, por ejemplo, el escenario de un servicio de transmisión de vídeo con clientes heterogéneos, donde los múltiples flujos de contenido de la fuente difieren en tamaño de la imagen, frame rate, y la tasa de bits. Con SVC, el contenido de la fuente tiene que ser codificado una sola vez, obteniendo un flujo de bits escalable, donde las representaciones de menor resolución y/o la calidad se puede obtener descartando los datos seleccionados. Por tanto, la necesidad de transcodificar los contenidos en términos de adaptación ya no es necesaria, simplificando así el proceso para reaccionar inmediatamente ante cambios en las condiciones de red.

SVC ofrece muchos beneficios en las aplicaciones de vídeo. Sin embargo, el despliegue de SVC presenta una serie de problemas relacionados principalmente con la estandarización y compatibilidad. Uno de los principales obstáculos del desarrollo son los equipos que no presentan compatibilidad con SVC. Como se ha comentado, la capa base de SVC se puede encapsular en un stream de una única capa con AVC, pero éste sería un stream de una calidad muy baja ya que representaría el nivel de escalabilidad disponible más bajo. Esto significa que los equipos que no soporten la tecnología SVC serán capaces de recibir únicamente la resolución más baja, el *frame rate* más bajo y el *bit rate* menor de la versión del stream.

5.1.2.1. SVC en detalle

SVC se estandarizó como una extensión de H.264/AVC, heredando su estructura y manteniendo los conceptos de *Video Coding Layer* (VCL) y *Network Abstraction Layer* (NAL) para lograr la independencia de la aplicación y de la red. Por un lado, la capa VCL se define para la codificación del vídeo, y la capa NAL convierte la representación VCL del vídeo en el formato correcto en función del medio de transporte o almacenamiento que se vaya a usar.

Por otro lado, para manejar el *bitstream* de una manera sencilla y sin necesidad de analizar la parte de vídeo codificado, se añaden 3 bytes a la cabecera de las unidades NAL, con información adicional que permite llevar a cabo los procesos de adaptación. Destacamos los identificadores DID (*Dependency ID*), QID (*Quality ID*) y TID (*Temporal ID*) que permiten establecer la asociación de la unidad NAL con la capa de escalabilidad a la que pertenece.

Los detalles referentes a la descripción de las estructuras VCL y NAL de la codificación H.264/SVC y la descripción de los niveles de escalabilidad se discuten a continuación. Para más información sobre el formato del fichero SVC, consultar la referencia (Amon et. al, 2007).

Niveles de escalabilidad

Decimos que un *stream* es escalable cuando podemos eliminar partes del mismo, dando como resultado un *sub-stream* que también es válido y que representa los contenidos originales pero con una resolución y/o *frame rate* y/o *bitrate* menor que el *stream* completo de la fuente original.

SVC permite 3 tipos de escalabilidad: espacial, temporal y de calidad. Con esto se persigue servir vídeo a diferentes clientes, que acceden con diferentes dispositivos conectados a través de diferentes redes con una única versión codificada del vídeo.

- Escalabilidad espacial: se refiere al tamaño de la imagen o resolución.
- Escalabilidad temporal: *frame rate*. La escalabilidad temporal ya estaba contemplada en H.264/AVC. SVC únicamente perfecciona su uso proporcionando información suplementaria de mejora.
- Escalabilidad de calidad: se consiguen *sub-streams* con la misma resolución espacial y temporal, pero con una SNR (*Signal to Noise Ratio*) menor que el *stream* original.

Evidentemente, las diferentes opciones de escalabilidad se pueden combinar, obteniendo así diferentes opciones espacio-temporales y *bit rates*. De hecho, un vídeo codificado con SVC está formado por una capa base y una o varias capas de mejora. La capa base estará formada por el *sub-stream* de mínima resolución, mínimo frame rate y mínima calidad. El resto de capas suponen una mejora añadida, bien sea en calidad, en resolución o en *frame rate*.

Escalabilidad temporal

La escalabilidad temporal es posible gracias al empleo de una estructura de predicción jerárquica, usando *frames* tipo B. Los slices tipo B se usan normalmente para las capas de mejora temporales debido a su propiedad *no-reference*. Esto es, imágenes B que dependen de imágenes B hacen que éstas no sean necesarias para ninguna otra imagen en el flujo de datos.

Podemos realizar escalabilidad temporal eliminando NALUs con un TID mayor que un valor dado. Por ejemplo, si identificamos las capas temporales con T (empezando en 0 para la capa base e incrementando de uno en uno para la siguiente capa temporal), entonces, para cada número natural k, el *bitstream* que se obtiene al eliminar todas las NALUs con capas temporales con un identificador T>k forma un nuevo *bitstream* válido. Para cada capa eliminada de un nivel superior de escalabilidad temporal, el *bitstream* resultante tiene una resolución temporal reducida en un factor 2.

La escalabilidad temporal es una ventaja frente a los codificadores de una sola capa, ya que introduce una estructura de dependencia de *frames* que permite decodificar el *bitstream* incluso cuando alguno de los datos se pierden. Es, por tanto, un mecanismo de resistencia frente a errores, ya que es posible decodificar los datos incluso cuando parte de ellos se pierden.

En la Figura 5.2 se aprecia que, si los *frames* que se corresponden con los números impares se pierden, el resto del *stream* no se ve gravemente afectado. Si, por ejemplo, se perdiese el frame 6, únicamente se vería afectado otro frame. Por tanto, esta escalabilidad temporal es mucho más resistente a errores que un *stream* de una única capa. De hecho, los *streams* codificados con una sola capa no ofrecen una buena experiencia de usuario (QoE, *Quality of Experience*) en redes que presentan más de un 1 % de paquetes perdidos o, en caso de emplear esquemas sofisticados de corrección de errores, hasta un 5 %. Sin embargo, con SVC, algunos fabricantes hablan de calidad de reproducciones aceptable

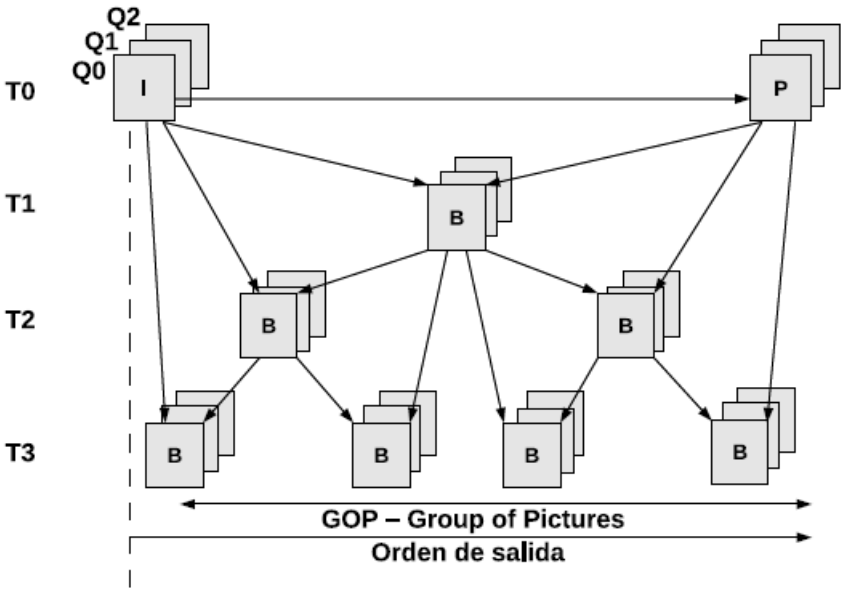


Figura 5.2: Ejemplo de escalabilidad temporal

incluso con pérdidas de hasta un 40 % en la red.

Escalabilidad espacial

La idea del escalado en la dimensión espacial reside en la codificación de las imágenes de distinta resolución en diferentes NALUs VCL. De esta manera, la unidad VCL que contiene la información extra para decodificar la imagen a mayor resolución se puede descartar si queremos reducir el escalado espacial.

Para lograr mejorar la eficiencia de codificación, SVC introduce el mecanismo de «predicción inter-capa» («*inter-layer prediction*»). El objetivo de las técnicas de predicción inter-capa es explotar la redundancia que existe entre las diferentes capas, con el fin de mejorar la eficiencia de codificación en las capas de mejora.

El identificador de dependencia para la capa base adquiere el valor 0, y se incrementa en una unidad entre una capa espacial y la siguiente. Eliminando capas cuyo DID sea mayor que uno dado conseguimos reducir la escalabilidad espacial.

Escalabilidad de calidad

Se puede considerar como un caso especial de escalabilidad espacial con imágenes de igual tamaño para la capa base y las de mejora.

Para la escalabilidad de calidad existen dos posibilidades: CGS (*Coarse-Grain quality scalable*) y MGS (*Medium-Grain quality scalable*).

- CGS: escalabilidad de calidad tradicional, donde se incrementa el número de bits por píxel para mejorar la calidad.

- MGS: se realiza de forma similar a la escalabilidad CGS en lo referente a que sólo contempla mejora de calidad. La diferencia está en que las unidades de datos MGS se pueden desechar sin afectar al resultado del *bitstream*, mientras que en el tipo de escalabildiad CGS, todas las unidades de datos (o *slices*) de la capa completa se tienen que procesar o se tienen que descartar. Por tanto, los *slices* codificados con el tipo de escalabilidad MGS se pueden eliminar individualmente, esto es, no es necesario eliminar la capa completa. El resultado es que MGS proporciona mayor eficiencia de codificación porque se puede descartar cualquier unidad individual de las capas de mejora.

Las capas de calidad reducen el error de cuantificación de codificación. De igual forma que en los casos de escalabilidad anteriores, el escalado de calidad se puede realizar eliminando NALUs con un QID mayor de un valor especificado.

Video Coding Layer (VCL)

VCL es el interfaz entre el encoder y los *frames* de vídeo. Al igual que otros códecs de vídeo, un vídeo codificado en SVC consiste en una secuencia de imágenes, estos son las *Access Unit* (AU), que contienen la información necesaria para decodificar exactamente una imagen. Cada *Access Unit* se divide en *slices* codificados. Tanto en SVC como en AVC hay tres tipos de *slices* y SVC adopta los conceptos de I, P, B sobradamente conocidos (ver sección 5.1.1).

En VCL se emplean diferentes técnicas para conseguir la escalabilidad espacial, temporal y de calidad. Por ejemplo, para la escalabilidad espacial se usa un filtro de sub-muestreo que genera una señal de resolución más baja para cada capa espacial. Para la escalabilidad temporal se emplean imágenes jerárquicas tipo B o un filtro temporal de compensación de movimiento (MCTF) para crear una descomposición temporal en cada capa espacial que permita la escalabilidad temporal.

Sin embargo, para decidir qué slice codificado pertenece a qué capa (bien sea temporal, espacial o de calidad), es neceario añadir información adicional al *slice*. Éste es el objetivo de la parte NAL (*Network Abstraction Layer*).

Network Abstraction Layer (NAL)

NAL es la interfaz entre el encoder y el protocolo de red que se empleará para transmitir el *bitstream*. El NAL encoder encapsula los *slices,* que proceden de la salida del VCL, en unidades NAL (NALU). Las NALU son las unidades fundamentales tanto en el *bitstream* de H.264/AVC como en H.264/SVC y permiten la transmisión de los datos de vídeo. Para generar de manera adecuada las unidades NAL, tenemos que definir el protocolo de red que emplearemos para la transmisión, siendo el más utilizado el consistente en empaquetar unidades NAL en paquetes RTP.

La sintaxis de la unidad NAL consta de una cabecera y una carga útil (ver Figura 5.3). La cabecera varía en función del tipo de codificación H.264 empleado: para AVC, la cabecera de la unidad NAL consta de 1 byte, mientras que para el caso de SVC se añaden 3 bytes extra para identificar a qué capa temporal, espacial o de calidad pertenece dicho

slice. La carga útil o *payload* de las unidades NAL lo constituyen los datos VCL, de forma que para cada *frame* de una resolución espacio-temporal dada, los datos se encapsulan en una o más unidades NAL.

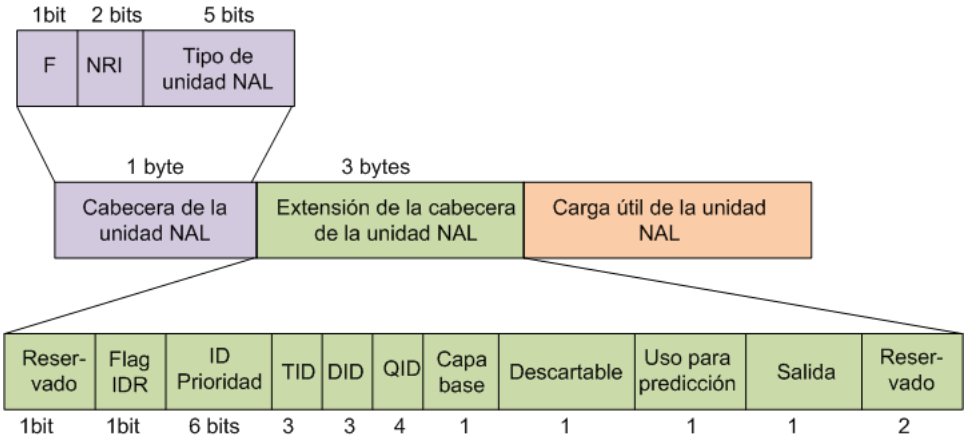


Figura 5.3: Estructura de la unidad NAL SVC

El primer byte de la cabecera NAL SVC comparte sintaxis con H.264/AVC, donde se indica el tipo de unidad NAL, la posible presencia de errores en la carga e información relativa a la importancia de dicha unidad NAL a la hora de realizar el proceso de decodificación (ver Figura 5.3). Ciertos valores del campo «tipo de unidad NAL» se asocian con la extensión escalable. Concretamente, los tipos correspondientes a los valores 14, 15 y 20 son empleados por H.264/SVC para identificar 3 nuevos tipos de NALUs. Las unidades NAL tipo 14 identifican a los "*prefix NAL unit*", las unidades NAL tipo 15 se emplean para los "*sequence parameter sets*" y las unidades tipo 20 se usan para los "*codec slices in scalable extension*". Estos tipos de unidad NAL indican la presencia de 3 bytes adicionales en la cabecera y proporcionan la información de dependencia de las capas de la extensión SVC.

Las unidades NAL correspondientes a *Sequence Parameter Se*t (SPS) presentan información en su carga útil referente a parámetros de codificación comunes a un determinado número de *slices.* Las unidades NAL tipo 20 indican la presencia de vídeo codificado en la extensión escalable en la parte de la carga útil. Las unidades NAL tipo 14 se emplean como prefijos para las unidades correspondientes a las NALUs de la capa base, compatibles con el codec H.264/AVC y, por tanto, carentes de la información descriptiva de escalabilidad.

Para los mencionados 3 octetos de cabecera extra de SVC, se emplea la sintaxis de los elementos de la cabecera de *priority_ id*, *dependency_ id*, *temporal_ id* y *quality_ id* para señalizar la información de escalabilidad dentro de cada unidad NAL. También se añade información acerca de la posibilidad de descartar una cierta unidad NAL. Toda esta información en la cabecera de las unidades NAL SVC permite identificar la asociación de cada unidad NAL SVC a su correspondiente capa de mejora sin la necesidad de analizar sintácticamente la parte del *payload* de la unidad NAL SVC. El campo TID (*temporal_ id*) indica la jerarquía entre capas temporales para la escalabilidad temporal; DID (*dependency_ id*) se emplea para la escalabilidad espacial, denotando la dependencia de codificación inter-capa entre las capas de mejora más altas/bajas; y QID (*quality_ id*)

designa la jerarquía de niveles de calidad para la escalabilidad de calidad MGS (*Medium Grain Scalability*).

5.2. Tráfico que generan los frames y efecto de la pérdida de frames

Puesto que el objetivo final consiste en la transmisión de los contenidos, es importante hacer notar cómo es el tráfico que generan los *frames* en los procesos de transmisión a a través de una red de datos y cómo afectan las pérdidas de *frames* a la calidad recibida.

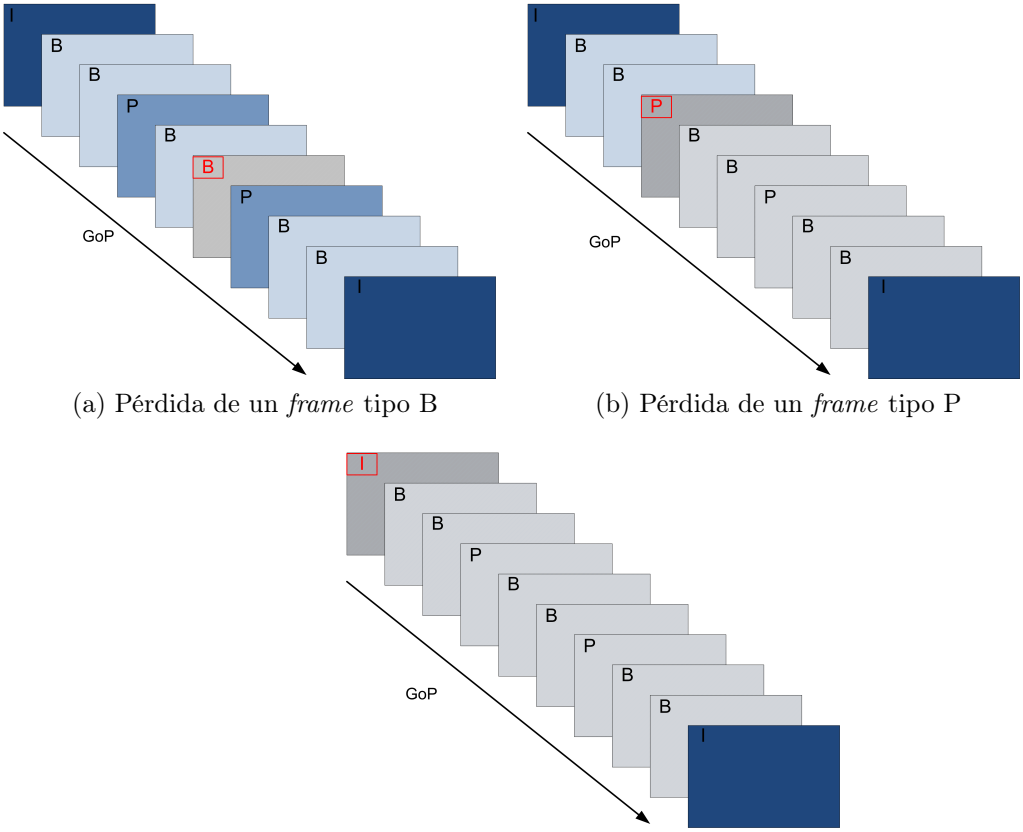
Hemos comentado que existen diferentes tipos de *frames*. La cantidad de información que es necesario transmitir por cada *frame* es variable. Por ejemplo, los *frames* tipo I contienen mucha más información que el resto, por lo que, a la hora de empaquetar la información para su transporte, ocuparán un mayor número de paquetes. Por el contrario, los *frames* tipo P o B sufren mayores niveles de compresión, así que el número de paquetes necesario para transmitir dichos *frames* será mucho menor.

Además del tipo de *frame*, otro factor que influye en el tráfico generado es el tipo de contenidos. Los vídeos que presentan poco movimiento necesitarán menos *frames* tipo I, ya que *frames* tipo P o B (que se codifican en base a diferencias con los anteriores) pueden resultar suficiente. Sin embargo, si la escena presenta muchos cambios y movimiento, será necesario emplear mayor número de *frames* tipo I, con lo que el tráfico generado presentará un volumen mayor.

La pérdida de paquetes durante la transmisión es un factor clave. El impacto visual de la pérdida de paquetes en la calidad del vídeo que se envía depende de las características del procesado de las pérdidas en la red y en el destino, pero sobre todo depende del esquema de codificación que se usa en el origen. Con algoritmos de compresión como pueden ser los empleados en la codificación MPEG-4, la pérdida de un único paquete puede degradar la calidad del vídeo ya que los errores a la hora de decodificar se transmiten hasta que se vuelva a recibir otra imagen completa de referencia. En este sentido, el tipo de *frame* afectado por las pérdidas es determinante: si se pierde un *frame* tipo B no tendrá mayor repercusión en los otros *frames* (Figura 5.4a). En cambio, si la pérdida afecta a un *frame* tipo P no se podrán decodificar ni *frames* P ni B en el mismo GoP (Figura 5.4b). Por último, en el peor de los casos, la pérdida de un I-*frame* se traduce en errores en la decodificación que afectan a todos *frames* del GoP (Figura 5.4c) (Kanakia et. al, 1995).

Algunos fabricantes implementan algoritmos de control de errores (FEC) para intentar solucionar este problema a base de información redundante. El problema está en que este tipo de soluciones suelen incrementar el ancho de banda necesario y la latencia ya que los códigos de control de error de bloques de paquetes se calculan antes de que éstos se envíen.

En el caso concreto del códec SVC, la pérdida de paquetes en las capas de mejora tienen un menor impacto, ya que están construidas sobre la capa base y no sobre frames previos. Esto quiere decir que si se pierden paquetes en la capa de mejora se traducirá en una pérdida temporal de resolución o frame rate, pero no en la estructura básica de



(a) Pérdida de un *frame* tipo B

(b) Pérdida de un *frame* tipo P

(c) Pérdida de un *frame* tipo I

Figura 5.4: Ejemplos de pérdidas de *frames*

la imagen de vídeo. Podríamos emplear FEC para proteger la capa base de la misma manera que se emplea en los codificadores no escalables. Otros vendedores, como el caso de la empresa RADVISION, emplean *Unequal Error Protection* (UEP) que protege cada nivel de diferente manera (los datos importantes, presentes en la capa base, se protegen más que los datos de las capas de mejora) (Radvision, 2013).

El contenido del vídeo también influye a la hora de visionar un vídeo con pérdidas. Por ejemplo, los "bloques" o el congelado de *frames* son más notables durante secuencias que contienen mucho movimiento. También hay estudios que sostienen que si las pérdidas ocurren al final del vídeo, la percepción de la calidad será menor que si las mismas pérdidas ocurren al inicio de la secuencia seguidas por un periodo de calidad relativamente buena (Liu et. al, 2007).

Capítulo 6

TECNOLOGÍA STREAMING

El consumo de contenidos multimedia a través de Internet genera un importante volumen de tráfico en la red y la tendencia sigue en aumento. Los servicios *streaming* se emplean hoy en día en periódicos o radios para transmitir eventos en directo, en páginas web donde el usuario puede acceder a los contenidos en directo o en diferido y, por supuesto, en videoconferencias y educación a distancia.

El conocimiento de las principales características de esta tecnología resulta clave en el desarrollo de los capítulos posteriores. El funcionamiento, arquitectura y protocolos implicados en la transmisión del contenido mediante *streaming* se resumen en las secciones sucesivas.

6.1. Introducción

La tecnología *streaming* nos permite la distribución de contenidos multimedia (esto es, audio y vídeo) a través de una red de datos (como puede ser Internet) de una manera continua y en tiempo real, sin la necesidad de descargar el archivo.

Surge a partir de la necesidad de solventar los problemas de la distribución multimedia a través de una red de datos basada en IP, en la que no se aplican técnicas específicas para este tipo de contenidos con necesidades de tiempo real. Hay que tener en cuenta que los contenidos de audio y vídeo difieren enormemente de los contenidos de datos en cuanto a los requisitos de transmisión, ya que necesitan ser reproducidos de manera continua. Por tanto, era necesaria una nueva tecnología que llevase a cabo el proceso de adaptación de los contenidos a las características de las redes IP, de manera que fuese posible la distribución de contenidos de naturaleza multimedia.

Los procesos de transmisión de vídeo *streaming* tradicionales exigen la presencia de dos flujos o canales diferenciados: por un lado el flujo de control y por otro lado el flujo para el envío de los datos multimedia (Figura 6.1). El primero de ellos se emplea para gestionar el establecimiento de sesión, así como la comunicación y las interacciones que el usuario pueda hacer con el servidor. Por otro lado, en el canal de transporte se envían los paquetes de contenido multimedia propiamente dicho.

Tanto el cliente como el servidor llevan a cabo una serie de trámites e intercambio

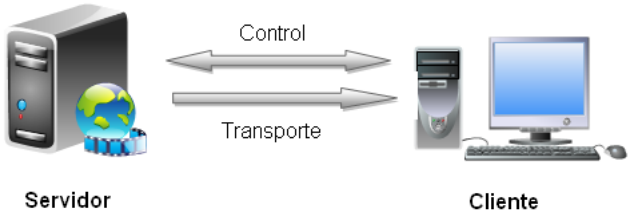


Figura 6.1: Flujo de control y de transporte en una sesión multimedia

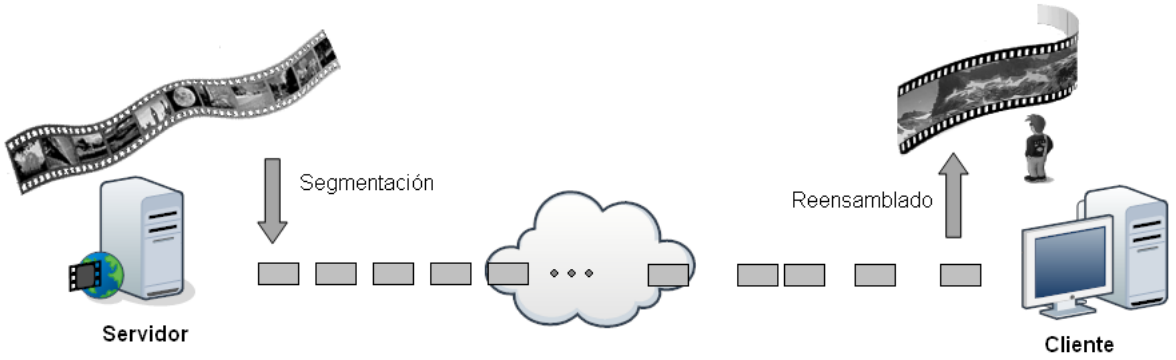


Figura 6.2: *Streaming*

de mensajes para establecer la sesión. Una vez establecida la sesión, puede comenzar la transmisión de los contenidos, que tienen que sufrir un proceso de adaptación para poder ser enviados a través de una red de datos. Esta adaptación consiste en la segmentación de la información multimedia en paquetes de un tamaño adecuado para su distribución a través de Internet. El objetivo de esta segmentación es la adaptación de los datos de vídeo (que tienen requisitos en tiempo real) a una filosofía de transmisión de conmutación de paquetes. Además, se añade información para mantener el control sobre el orden y los instantes de transmisión o las pérdidas que se puedan ocasionar, puesto que es posible que cada paquete viaje por un camino diferente, con diferentes condiciones de transmisión. La integridad de los paquetes recibidos se podrá analizar consultando la información de las cabeceras, pudiendo así reensamblar el contenido recibido para su reproducción. La Figura 6.2 ejemplifica el proceso descrito.

A medida que el cliente recibe los paquetes de contenidos multimedia, procede a su almacenamiento en un *buffer* (Figura 6.3). Cuando el *buffer* se llena el cliente comienza la reproducción, mientras que al mismo tiempo continúa recibiendo nuevos paquetes de información. Este *buffer* del cliente también permite amortiguar los efectos de pequeñas variaciones en las condiciones de la red, de manera que si se produce un descenso en el ancho de banda disponible para la transmisión, el cliente puede continuar con la reproducción de los paquetes almacenados en el *buffer*. Sin embargo, si los problemas de disponibilidad de ancho de banda persisten, dicho *buffer* se vaciará y por tanto la reproducción se detendrá. Unas condiciones de transmisión estables son la situación ideal desde el punto de vista de la calidad percibida por parte del usuario durante la reproducción.

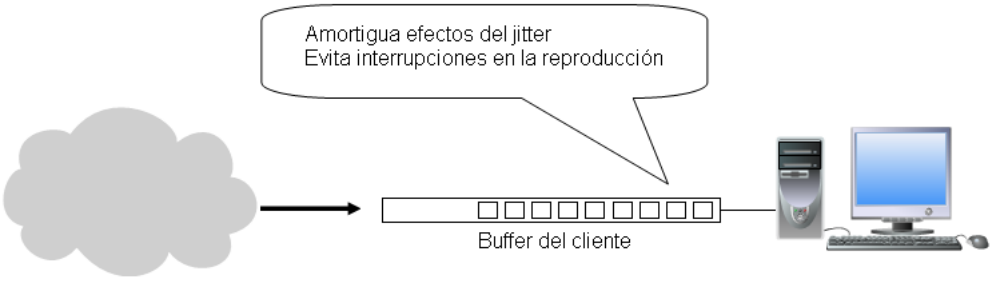


Figura 6.3: *Buffer* del cliente

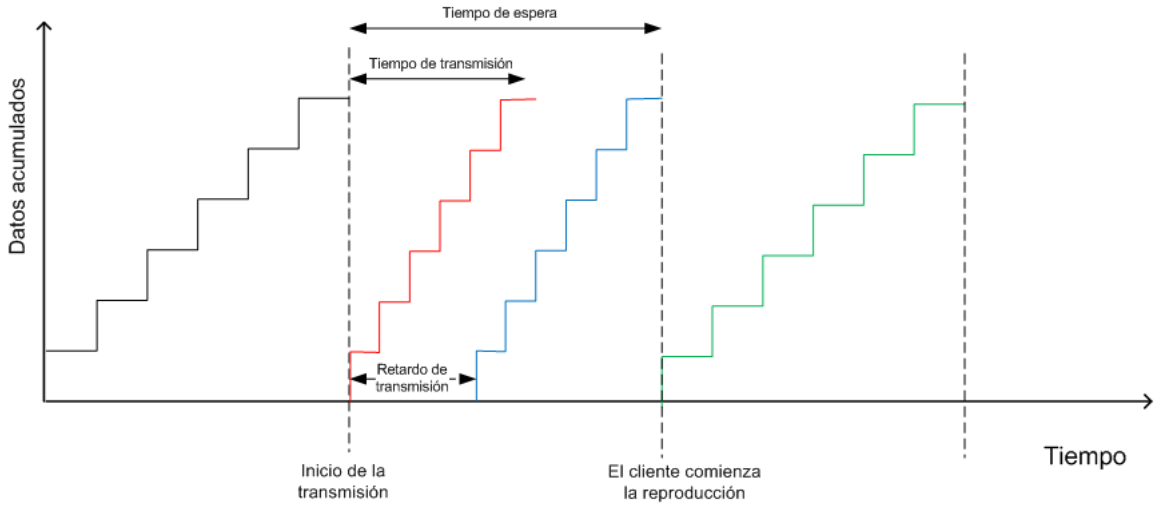


Figura 6.4: Tiempo de espera para un modelo de descarga y reproducción

La principal ventaja de este sistema es la visualización del contenido a la vez que se descarga (prácticamente en tiempo real), sin que sea necesario almacenar previamente los datos que queremos reproducir. En los procesos de descarga tradicional, eran necesarias grandes esperas y suficiente espacio libre en el cliente para almacenar el archivo antes de poder reproducirlo.

Existe una solución intermedia denominada *descarga progresiva*. En la descarga progresiva, el cliente reproduce la información multimedia a medida que se lleva a cabo la descarga al disco o a la memoria, permitiendo reproducir los contenidos antes de que finalice la descarga, pero sin descartar los paquetes a medida que se reproducen. Esta combinación, a caballo entre la descarga tradicional y la tecnología *streaming*, era empleada en algunos servicios de vídeos *online* como *YouTube*.

En los siguientes esquemas (Figuras 6.4 y 6.5) ejemplificamos las diferencias, en lo referente a los tiempos de espera para un mismo contenido multimedia, entre el sistema tradicional de «*descarga+reproducción*» frente a la tecnología *streaming* .

En el caso de la descarga y su posterior reproducción (Figura 6.4), el tiempo transcurrido desde que el servidor comienza la transmisión hasta que el cliente reproduce los

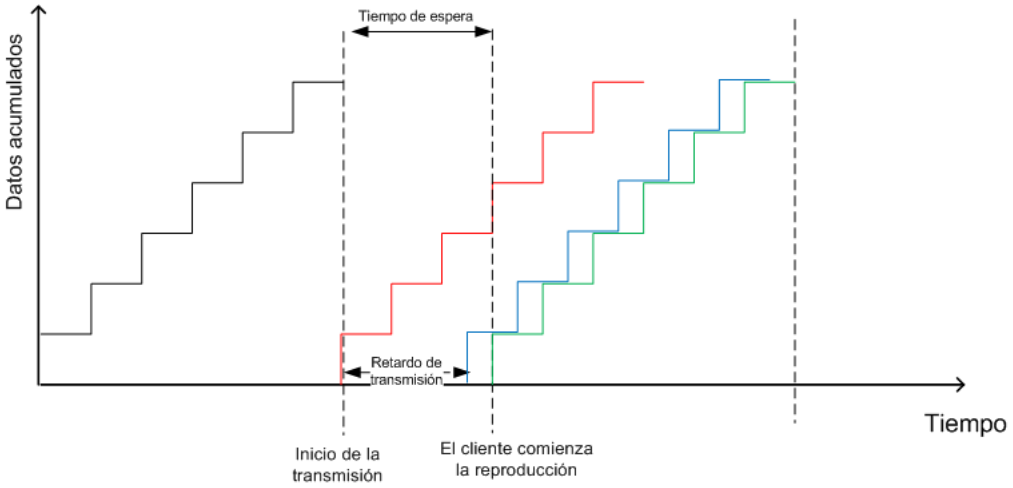


Figura 6.5: Tiempo de espera para un modelo de *streaming*

contenidos está compuesto por el tiempo de envío del archivo de audio/vídeo completo junto con el retardo propio del proceso de transmisión. Para el caso de la tecnología *streaming* (Figura 6.5), el retardo de transmisión seguirá estando presente, pero el cliente comenzará la reproducción poco después de haber recibido los primeros paquetes, mientras el servidor continúa transmitiendo el resto. Por otro lado, en el primer caso (Figura 6.4), la tasa de envío del contenido multimedia puede ser mayor que su velocidad de reproducción. Sin embargo, para el caso de la tecnología *streaming* (Figura 6.5), la tasa de transmisión será la misma que la de reproducción.

La tecnología *streaming* permite además realizar interacciones por parte del usuario con los contenidos recibidos. Así, por ejemplo, se contempla la realización de avances, retrocesos o pausas a lo largo de la línea temporal de la reproducción. La posibilidad de realizar alguna o todas las interacciones dependerá del tipo de servicio *streaming*. Existen dos tipos de modalidades a la hora de implementar un servicio de *streaming*: servicios bajo demanda y servicios en directo. También podríamos incluir un tercer tipo: el casi bajo demanda, donde se simula un sistema bajo demanda pero con flujos de vídeo en directo.

6.1.1. Servicios en directo

Basan su filosofía de funcionamiento en el concepto de multidifusión, de la misma forma que la emisión de la televisión o la radio: los usuarios acceden a un único flujo de información, en cualquier momento, pero teniendo en cuenta que todos ellos acceden a los mismos contenidos que se correspondan con el instante temporal en el que se encuentre la emisión.

En este caso, la única interacción permitida es la pausa. Si un usuario abandona el servicio o realiza una pausa, la emisión de los contenidos no se detiene, por lo que cuando el usuario decida retomar la reproducción, se comenzará en el punto donde se encuentre en ese momento la emisión (Figura 6.6).

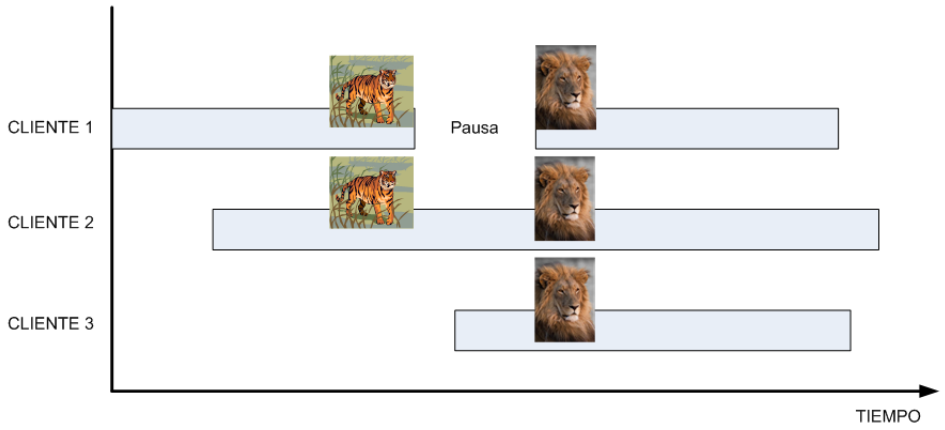


Figura 6.6: Servicio en directo

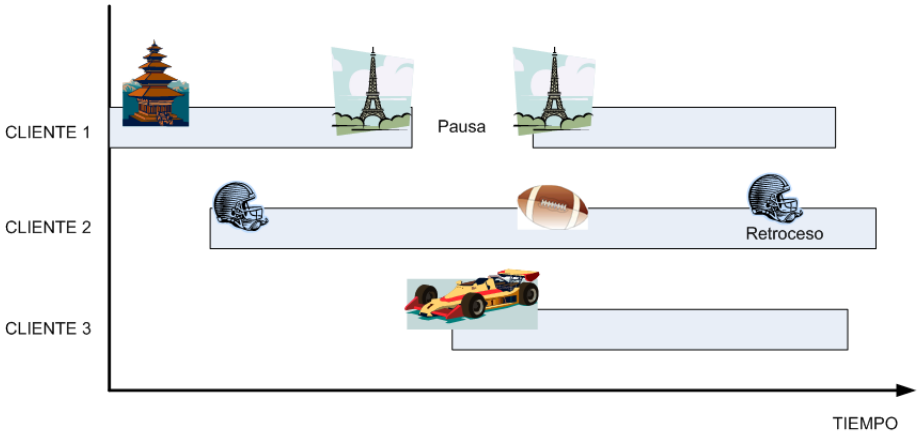


Figura 6.7: Servicio bajo demanda

Dentro de los contenidos en directo también es posible establecer una clasificación en función de los contenidos, bien sean contenidos almacenados previamente o contenidos que se emiten en vivo.

6.1.2. Servicios bajo demanda

A diferencia de los servicios en directo, los servicios bajo demanda envían el contenido que solicite el cliente en cualquier instante (Figura 6.7). En otras palabras, los contenidos están a disposición del cliente en el momento en que éste lo solicite.

En esta modalidad del servicio *streaming*, entre cliente y servidor se establece una conexión en la que el cliente dispone de un canal de información en exclusiva. Esto facilita el hecho de que el cliente pueda realizar mayor variedad de interacciones con el servidor, a diferencia de los servicios en directo. El usuario podría comenzar la reproducción en cualquier momento, realizar pausas, avances o retrocesos, de manera que el servidor transmite los contenidos desde el instante temporal solicitado por el cliente.

Los servicios bajo demanda únicamente pueden trabajar con contenidos almacenados.

6.2. Arquitectura

La arquitectura básica en la que se basan los servicios de *streaming* es una arquitectura cliente/servidor. Sin embargo, según el tipo de contenidos ofertados, pueden aparecer otros elementos en el sistema.

En la Figura 6.8 vemos un esquema de la arquitectura con todos los elementos que pueden estar presentes. En los sucesivos apartados explicaremos brevemente en qué consiste cada uno.

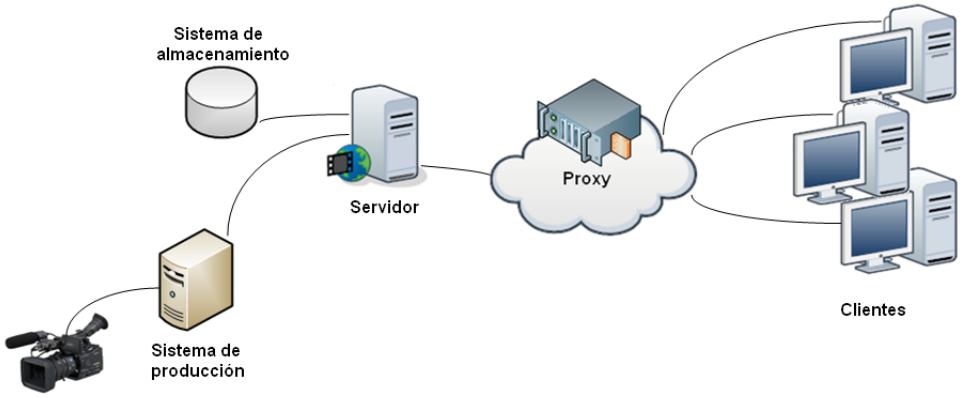


Figura 6.8: Arquitectura

6.2.1. Sistema de producción

El sistema de producción captura los contenidos en vivo de vídeo y audio y los transmite, en tiempo real y con el formato adecuado, al servidor.

La captura de los contenidos se hace a partir de elementos tales como micrófonos o cámaras. Posteriormente, los elementos software o hardware que componen el sistema de producción se encargarán de comprimir los contenidos y darles un formato adecuado, de manera que resulten aptos para su transmisión por medio de *streaming* sobre un protocolo de transporte, como RTP sobre UDP o sobre TCP (*Transmission Control Protocol*). Estas modificaciones de los contenidos originales abarcan desde los códecs de audio/vídeo a los cambios en las tasas de codificación.

Si estamos hablando de servicios de vídeo *streaming* bajo demanda, este procesamiento se puede hacer fuera de la arquitectura. Sin embargo, en los servicios en directo con contenidos en vivo este sistema tiene que formar parte de la arquitectura ya que el formato crudo en el que se capturan los contenidos se tiene que comprimir antes de poder transmitirlo.

6.2.2. Sistema de almacenamiento

Es el elemento encargado de almacenar los contenidos, de manera que puedan ser solicitados por los clientes en cualquier momento (en el caso del vídeo bajo demanda) o

que puedan ser transmitidos por el servidor en cualquier instante (servicio en directo con contenidos almacenados).

No sería necesario para un servicio *streaming* en directo con contenidos en vivo.

6.2.3. Servidor

Este elemento, a diferencia de los anteriores, no puede faltar en cualquier tipo de sistema de distribución de audio/vídeo, ya que tiene uno de los roles más importantes: su misión es recibir y procesar peticiones de los clientes, establecer la sesión mediante el intercambio de una serie de parámetros y procesar el envío de los contenidos multimedia, que se pueden encontrar en un sistema de almacenamiento o los puede recibir directamente del sistema de producción.

Como veremos en el apartado 6.3, se necesitan algunos protocolos para regular la transmisión de los paquetes, como por ejemplo un protocolo de transporte de capa baja como UDP (*User Datagram Protocol*) o a un nivel más alto como RTP (*Real-time Transport Protocol*), en combinación con RTCP (*Real Time Control Protocol*) y RTSP (*Real Time Streaming Protocol*).

Entre el servidor y el cliente se establecerán los canales de comunicación. El número de canales dependerá de los flujos que tenga la información que se transmite, esto es, si el contenido está formado por un flujo de audio y uno de vídeo, se establecerán dos canales de comunicación entre el cliente y el servidor.

6.2.4. Reproductor

El reproductor es el elemento del cliente que se encarga de la interacción con el servidor: establecer las conexiones, procesar los paquetes y reconstruir el flujo de datos recibido son parte de su funcionalidad.

En algunos casos, el reproductor puede intercambiar información con el servidor referente a estadísticas de transmisión y recepción de los datos.

6.2.5. Proxy

Este elemento puede aparecer en algunas configuraciones. Su objetivo es similar a un *proxy* http: almacenamiento temporal para disminuir el tráfico hacia la red exterior, con la salvedad de que en un *proxy* multimedia (donde la información es de tipo continuo) tiene que mantener la comunicación con el cliente durante la reproducción del vídeo, manteniendo el estado de la sesión e interactuando con el usuario (en el caso de un *proxy* http la conexión con el cliente sería puntual).

El almacenamiento temporal de los contenidos para evitar que los clientes accedan al servidor puede resultar especialmente útil en servicios bajo demanda para reducir el consumo de recursos tanto de red como los recursos computacionales del servidor.

Además del almacenamiento temporal, nos podemos encontrar con otro tipo de *proxys* funcionando en modo *pass-through*, que permiten realizar un balanceo de carga cuando hay más de un servidor de vídeo *streaming* disponible para acceder a los contenidos.

6.3. Protocolos

La aparición de la tecnología *streaming* ha sido posible gracias al desarrollo de una serie de protocolos que mejoraron y agilizaron notablemente el proceso de transmisión de datos. La aparición del protocolo UDP (*User Datagram Protocol*) frente a TCP (*Transmission Control Protocol*) permitió el envío de datagramas sin control de flujo ni necesidad de confirmación de entrega, evitando ralentizar el proceso con interrupciones cada vez que ocurría un error en el protocolo. El servicio orientado a conexión y fiable que ofrece TCP ocasionaría un retardo muy elevado, debido a los proceso de retransmisión, que los servicios en tiempo real no pueden asumir. Las aplicaciones que emplean UDP en la capa de transporte (como por ejemplo los servicios de *streaming*) asumen las pequeñas pérdidas que se pueden ocasionar, a cambio de conseguir un funcionamiento en tiempo real.

Pero además, fue necesario el desarrollo de otros protocolos como RTSP para el control de las sesiones, RTCP para controlar de alguna manera la calidad de la transmisión y RTP para el envío temporizado de la información.

6.3.1. RTSP

RTSP (*RealTime Streaming Protocol*) (RFC2326) es el protocolo encargado de mantener y controlar la sesión entre el cliente y el servidor de servicios multimedia basados en la tecnología *streaming*. El protocolo se centra únicamente en la parte de control del envío de datos en tiempo real, puesto que los datos propiamente dichos viajan por un canal diferente. Con RTSP se establece una comunicación entre cliente y servidor mediante la cual se pueden llevar a cabo interacciones, tales como las pausas, los avances o los retrocesos durante la reproducción. Digamos que actúa como un control remoto de los servidores multimedia. Para ello, cliente y servidor intercambian mensajes por medio de un conjunto de métodos, algunos de los cuales están descritos en la Tabla 6.1.

RTSP es independiente de la capa de transporte, y puede implementarse junto con protocolos no fiables de transporte y no orientados a conexión, como UDP, o protocolos fiables u orientados a conexión, como TCP (Figura 6.9).

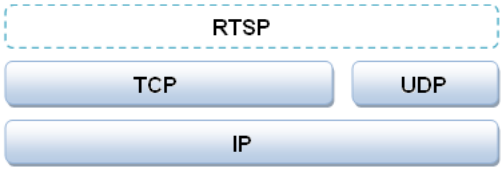


Figura 6.9: Pila de protocolos

Método	Descripción
OPTIONS	Con este mensaje el cliente solicita al servidor información sobre los métodos que soporta. Su uso no es obligatorio y formaría parte de un paso previo al establecimiento de la sesión.
DESCRIBE	La sesión la inicia el cliente con el comando RTSP DESCRIBE. Con este método el cliente obtiene una descripción de un objeto multimedia identificado por la URL (*Uniform Resource Locator*) de la petición. Dicha descripción contiene, por ejemplo, el número de flujos multimedia que necesitamos para la correcta reproducción de los contenidos, la codificación y otros parámetros de configuración de la sesión. Toda esta información que proporciona el servidor se incluye en un archivo SDP (*Session Description Protocol*).
SETUP	Se emplea para especificar información sobre las sesiones de transporte y control entre el cliente y el servidor, tales como números de puerto o direcciones IP. Al recibir este mensaje, el servidor reserva los recursos necesarios para comenzar la transmisión.
PLAY	Cuando el servidor reciba esta petición comenzará el envío de datos a través de los puertos especificados durante el intercambio de mensajes SETUP.
PAUSE	Se detiene temporalmente el envío de datos, pero no se liberan los recursos de la sesión.
TEARDOWN	Detiene el envío de datos y libera todos los recursos reservados para esa sesión.

Tabla 6.1: Métodos en RTSP

Otro aspecto a destacar de RTSP es que es un protocolo con estados. Distinguimos 3 estados:

- Inicial.
- Listo: estado posterior a la inicialización de la sesión.
- Emitiendo: estado en el que se lleva a cabo la distribución de los contenidos.

Mediante la ejecución de los métodos vistos en la Tabla 6.1 se lleva a cabo la transición entre un estado u otro (Figura 6.10). Por ejemplo, mediante el mensaje SETUP (en la fase de inicialización y con el que se obtiene información sobre la sesión de transporte) se genera la transición del estado INICIAL al estado LISTO. Una vez en el estado LISTO, el envío del contenido puede comenzar en cualquier instante a partir del método PLAY. Dicho envío se puede detener momentáneamente con el mensaje PAUSE, devolviendo la transmisión al estado LISTO, pudiendo reanudarse más adelante. Para liberar los recursos y volver al estado INICIAL enviaremos un mensaje del tipo TEARDOWN.

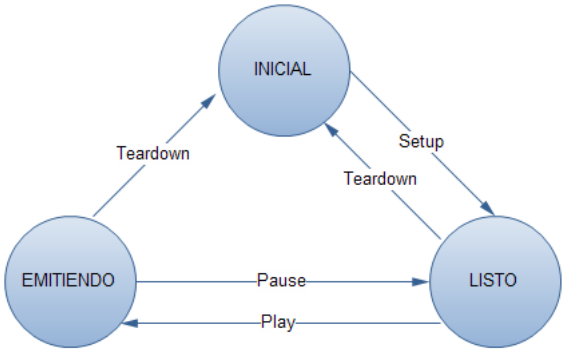


Figura 6.10: Diagrama de estados RTSP

Por tanto, para llevar a cabo un proceso de establecimiento de sesión, es necesario llevar a cabo el intercambio de una serie de mensajes entre el cliente y el servidor, tal y como se detalla en la Figura 6.11. En la citada figura, además del intercambio de mensajes propios del protocolo RTSP, también observamos el envío de la información multimedia mediante el protocolo RTP y el envío de información de control por parte del cliente por medio del protocolo RTCP (protocolos que se introducen en los siguientes apartados).

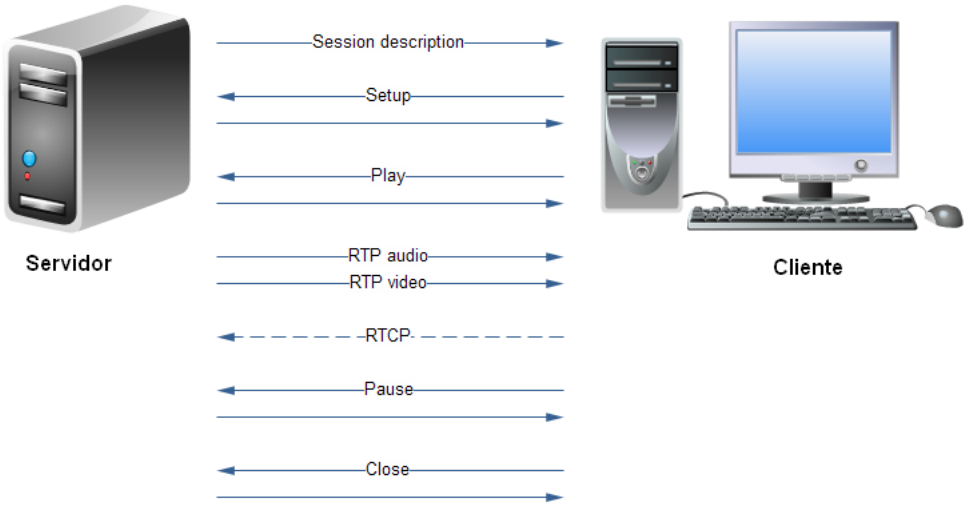


Figura 6.11: Establecimiento de conexión RTSP

6.3.2. RTP

RTP (*Real-time Transport Protocol*) (RFC3550) es el estándar para el transporte extremo a extremo de datos con características de tiempo real, como audio y vídeo, sobre redes IP. Asume la existencia de pérdidas y retardos, así como la posibilidad de variación de las características de la red en el transcurso de la comunicación.

El protocolo define el formato de los paquetes, que consta de una cabecera y la carga útil. La cabecera contiene la información necesaria para reconstruir el flujo de bits generado por el *codec* del emisor y también contiene información de sincronización. Esta

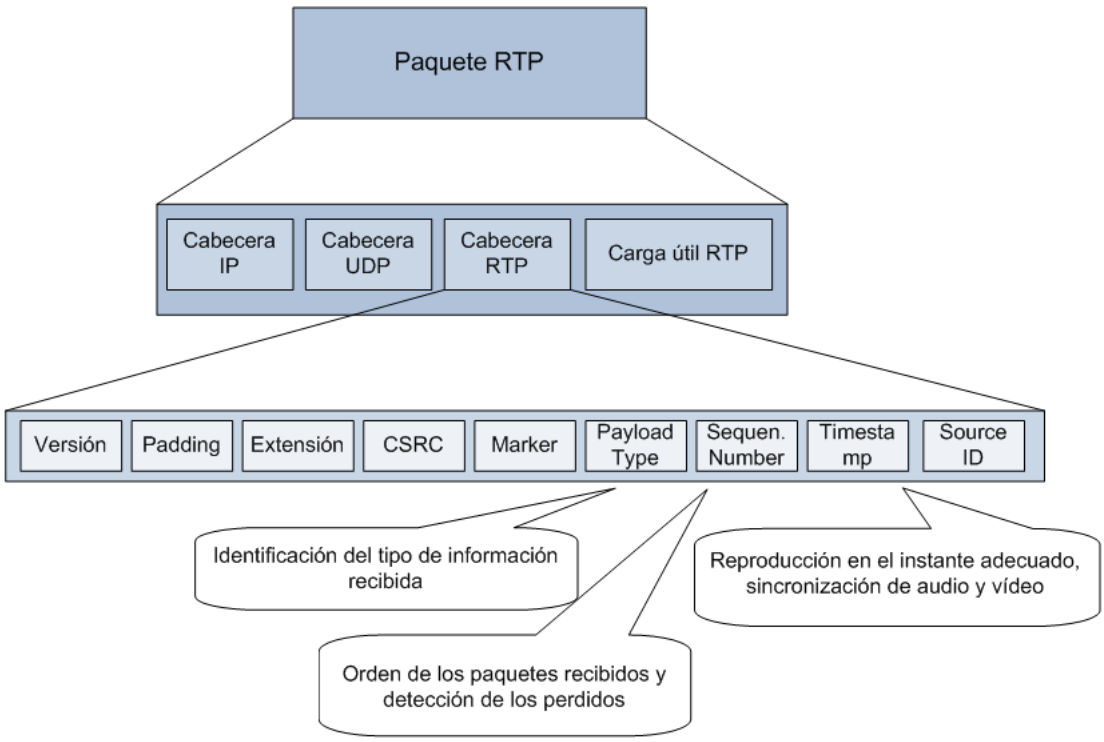


Figura 6.12: Paquete RTP

cabecera añade una sobrecarga de 12 bytes. RTP permite, además, añadir números de secuencia con el fin de ordenar los paquetes y detectar pérdidas.

En las Figuras 6.12 y 6.13 podemos ver la estructura del paquete y de la cabecera RTP (extraído del RFC3550) respectivamente, y en la Tabla 6.2 un resumen de los principales campos.

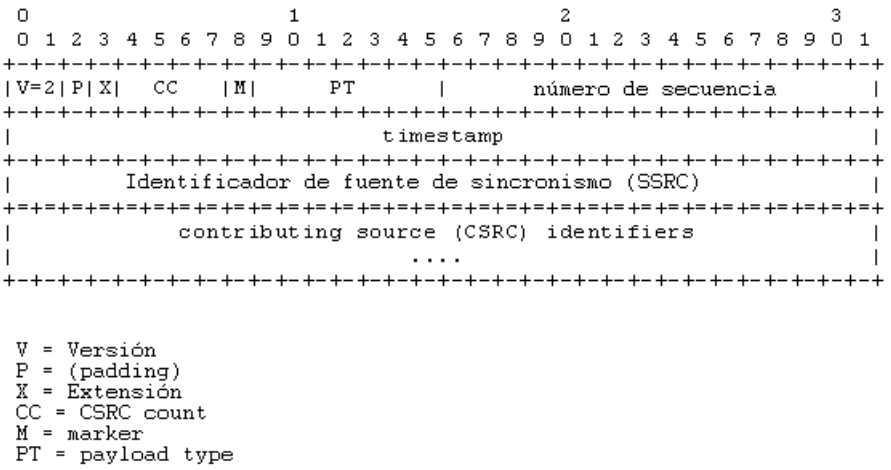


Figura 6.13: Cabecera RTP

Campos de la cabecera RTP	Sinopsis
Número de secuencia	Identificador de cada uno de los paquetes enviados por el servidor. El cliente lo emplea para reconstruir el orden de los contenidos recibidos. También se emplea para detectar pérdidas en la transmisión.
Timestamp	Indica el instante en el que se generó el paquete.
SSRC (*Synchronization Source*)	Identifica al elemento que realiza la multiplexación de fuentes.
CSRC (*Contribution Source*)	Identifica a la fuente de datos. Permite distinguir el origen de los datos si los contenidos se envían entremezclados.

Tabla 6.2: Algunos campos de la cabecera RTP

Generalmente RTP trabaja sobre UDP, aunque también podría emplearse TCP como protocolo subyacente en la capa de transporte.

RTP no dispone de ningún mecanismo para asegurar la calidad de servicio, ya que únicamente se centra en el transporte de los datos. Necesita apoyo de otros protocolos asociados, como RTCP.

6.3.3. RTCP

RTCP (*Real Time Control Protocol*) (RFC3550) se implementa en combinación con el protocolo RTP y se basa en la transmisión periódica de paquetes de control que monitorizan el envío de paquetes RTP. La información de control que se maneja se corresponde, principalmente, con la calidad de servicio (pérdida de paquetes, retardo, *jitter*, etc.). Esa información puede ser empleada por aplicaciones adaptativas para ajustar la codificación y otros parámetros. Aunque es opcional, se recomienda su utilización porque proporciona información de estado de la comunicación para detectar situaciones en las que la calidad no es suficiente. En ese caso, se podrán tomar las medidas oportunas y será la aplicación quien, en función de esa información, determine los mecanismos que se deberán emplear para mejorar la transmisión.

En la Tabla 6.3 se recogen algunos de los diferentes tipos de paquetes de control y sus características.

6.3.4. SDP

SDP (*Session Description Protocol*) (RFC4566) se emplea para describir sesiones multimedia en una gran variedad de aplicaciones, entre ellas destacan VoIP (*Voice over IP*) y *streaming*.

Con SDP se definen parámetros tales como los identificadores, los creadores de la sesión e incluso una descripción de los flujos que componen la sesión (nombre del flujo, tipo de

Paquetes RTCP	Tipo	Sinopsis
Informes de emisor (SR)	200	Proporciona información sobre los contenidos que están siendo transmitidos, principalmente para que los receptores puedan sincronizar múltiples streams.
Informes de receptor (RR)	201	Lo envían todos los participantes que están recibiendo datos y su objetivo es la recepción de informes de calidad.
Descriptores de fuente (SDES)	202	Identifica a los participantes de la sesión incluyendo su CNAME (nombre canónico) .
Mensaje de fin (BYE)	203	Se genera cuando un participante abandona la sesión
Específicos de la aplicación (APP)	204	Se pueden configurar a medida para cada aplicación. Por tanto, los datos que transmite son aquellos que se consideren oportunos.

Tabla 6.3: Tipos de paquetes RTCP

encriptación, etc.), así como direcciones IP, puertos y formatos necesarios para acceder a los medios, los protocolos de transporte empleados o el formato de la codificación.

6.4. Dynamic Adaptive Streaming over HTTP (DASH)

A pesar de que los protocolos *streaming* (ver sección 6.3) están diseñados con el propósito de efectuar de manera óptima el envío de información multimedia, el hecho innegable es que Internet está construido sobre HTTP y optimizado para el envío HTTP. Por tanto, la cuestión es ¿por qué no adaptar el envío multimedia a Internet, en lugar de proponer nuevos protocolos que adapten Internet al envío multimedia?

Cambiando el paradigma del *streaming* de audio/vídeo sobre RTP, surge una nueva generación de aplicaciones *streaming* basándose en HTTP (*Hypertext Transfer Protocol*). La tecnología recibe el nombre de DASH (*Dynamic Adaptive Streaming over HTTP*) (ISO/IEC, 2014) y ha sido recientemente estandarizada a finales de 2011. Previamente, diferentes plataformas comerciales empleaban *streaming* HTTP como método subyacente de envío. Sin embargo, era necesario estandarizar los protocolos de envío HTTP *streaming* con el fin de conseguir interoperabilidad entre diferentes implementaciones de servidores y clientes.

El uso de conexiones HTTP/TCP permite reutilizar la infraestructura de red existente, incluyendo las redes de distribución de contenidos (CDN, *Content Delivery Network*), lo que presenta enormes ventajas a la hora de efectuar el despliegue del servicio. Además, se cancelan de problemas típicos referidos a los *firewalls* y NAT con RTP/UDP, puesto que generalmente este tipo de paquetes no están permitidos en los *firewalls*. Por último se solucionan los problemas de disponibilidad de recursos a gran escala de los servidores basados en RTP, debido a la necesidad de mantener una sesión *streaming* para cada cliente. A todo esto hay que añadir las posibilidades que brinda DASH a la hora de confeccionar servicios *streaming* adaptativos que mejoren la QoE. Todas estas ventajas ofertadas, en términos de utilización de recursos y calidad percibida por el usuario, consiguen que el

streaming adaptativo sobre HTTP esté siendo gradualmente adoptado por los proveedores de contenidos y servicios en la red.

6.4.1. Introducción

DASH pretende solucionar las debilidades que puedan presentar las soluciones streaming basadas en RTP/RTSP y las soluciones basadas en descarga progresiva, permitiendo la adaptación de contenidos de vídeo *streaming* a través de Internet usando el protocolo HTTP y proporcionando servicios *streaming* para usuarios con condiciones de red dinámicas y dispositivos heterogéneos.

En DASH el contenido multimedia se codifica en múltiples versiones. Cada una de estas versiones (en adelante, representaciones) presenta una opción diferente en cuanto a los parámetros de codificación. Es decir, las diferentes representaciones del contenido multimedia difieren entre sí en valores tales como la tasa de codificación, la resolución espacial o la resolución temporal, dando lugar a diferentes niveles de calidad para los mismos contenidos. Pero además, cada una de las representaciones se divide a su vez en pequeños segmentos de vídeo que se envían al cliente a través de servidores HTTP estándar. Cada segmento tiene su propia URL y el reproductor es capaz de solicitar y reproducir cada segmento individualmente. Mediante este planteamiento, el cliente puede dotar al sistema de cierto grado de adaptación, solicitando los segmentos que más se adecúen al ancho de banda disponible (Figura 6.14) (Sodagar, 2011).

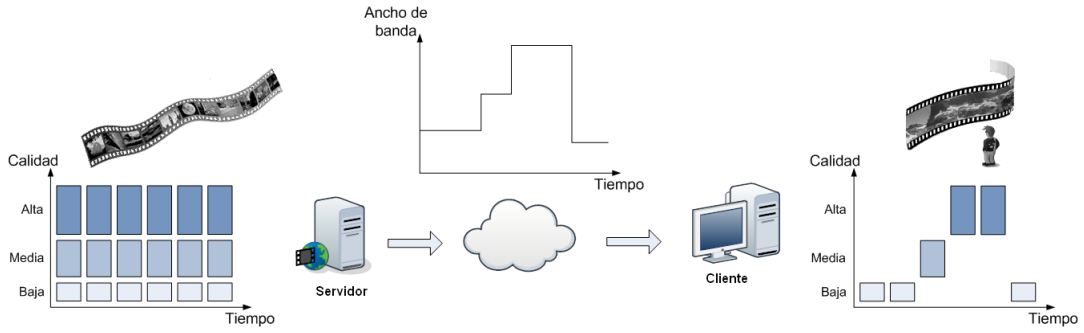


Figura 6.14: Escenario DASH

La descripción de las características de cada una de las representaciones de los contenidos y sus segmentos se especifica en un fichero XML (*eXtensible Markup Language*) denominado MPD (*Media Presentation Description*) y almacenado en el servidor HTTP junto con los contenidos.

Para reproducir los contenidos, el cliente primeramente deberá obtener el fichero MPD para, a continuación, extraer toda la información relevante para el proceso de reproducción. Una vez hecho esto, el cliente está en disposición de comenzar la transmisión multimedia mediante peticiones HTTP de los segmentos correspondientes a la representación de los medios que son acordes a sus características (Figura 6.15).

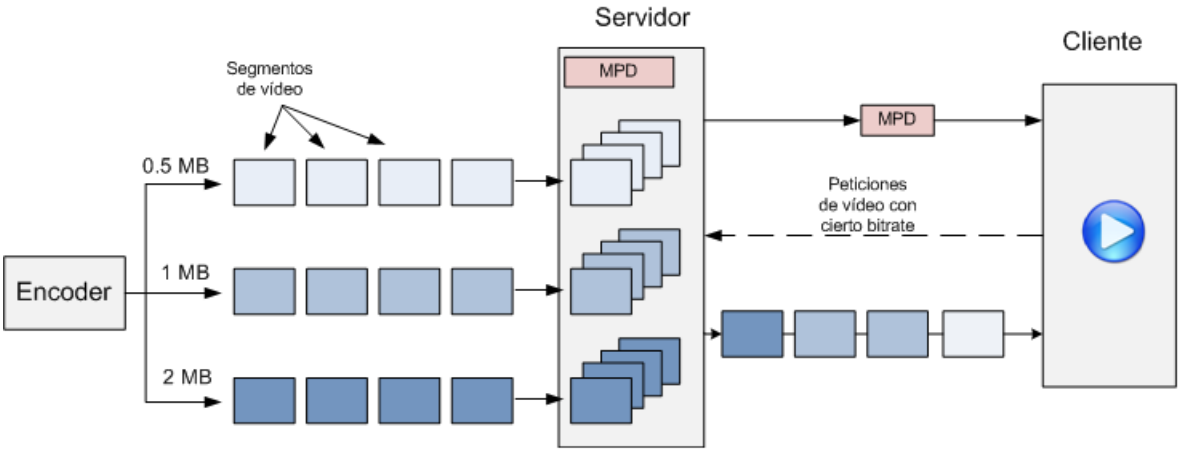


Figura 6.15: DASH estándar

6.4.1.1. El fichero MPD (Media Presentation Description)

El fichero MPD es un documento XML a partir del cual el cliente DASH obtiene la información necesaria para poder comenzar el servicio *streaming*. El fichero MPD sigue una estructura jerárquica, formada por:

- *Period*: se corresponde con un determinado intervalo temporal del contenido multimedia. Cada periodo tiene, por tanto, una duración determinada y está formado por uno o varios *adaptation sets*.

- *Adaptation set*: proporciona información acerca de las diferentes versiones de codificación de los contenidos. Por ejemplo, un *adaptation set* puede contener diferentes bitrates para los mismos contenidos multimedia. Otro *adaptation set* puede contener diferentes bitrates para los contenidos de audio. Cada *adaptation set* incluye diferentes *representations*.

- *Representation*: se corresponde con una alternativa en concreto de codificación. Las diferentes representaciones de los contenidos pueden diferir en el bitrate o la resolución, entre otras características. Cada *representation* consiste en uno o varios *segments*.

- *Segment*: cada una de las partes (o trozos) en las que se han dividido los contenidos. Cada segmento contiene su propia URL para que pueda ser descargado por medio de peticiones GET HTTP.

Gráficamente, la Figura 6.16 esquematiza la organización y la estructura de la información almacenada en el fichero MPD. Y un extracto del contenido real de este tipo de ficheros puede observarse en la Figura 6.17 (ejemplo obtenido del repositorio DASH [1]).

[1] http://www-itec.uni-klu.ac.at/dash/?page_id=207

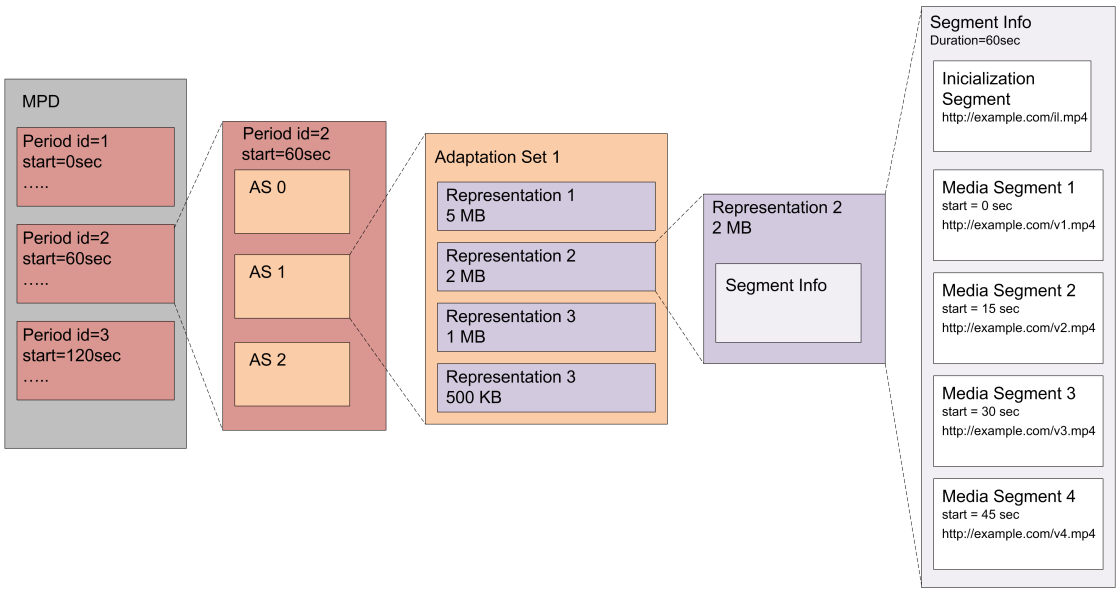


Figura 6.16: Ejemplo de jerarquía en el fichero MPD

```
▼<MPD xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="urn:mpeg:DASH:schema:MPD:2011"
 xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:2011" profiles="urn:mpeg:dash:profile:isoff-main:2011"
 type="static" mediaPresentationDuration="PT0H9M56.46S" minBufferTime="PT2.0S">
  ▼<BaseURL>
     http://www-itec.uni-klu.ac.at/ftp/datasets/mmsys12/BigBuckBunny/bunny_2s/
   </BaseURL>
  ▼<Period start="PT0S">
    ▼<AdaptationSet bitstreamSwitching="true">
      ▼<Representation id="0" codecs="avc1" mimeType="video/mp4" width="320" height="240"
       startWithSAP="1" bandwidth="45652">
        ▼<SegmentBase>
           <Initialization sourceURL="bunny_2s_50kbit/bunny_50kbit_dash.mp4"/>
         </SegmentBase>
        ▼<SegmentList duration="2">
           <SegmentURL media="bunny_2s_50kbit/bunny_2s1.m4s"/>
           <SegmentURL media="bunny_2s_50kbit/bunny_2s2.m4s"/>
           <SegmentURL media="bunny_2s_50kbit/bunny_2s3.m4s"/>
           <SegmentURL media="bunny_2s_50kbit/bunny_2s4.m4s"/>
           <SegmentURL media="bunny_2s_50kbit/bunny_2s5.m4s"/>
           <SegmentURL media="bunny_2s_50kbit/bunny_2s6.m4s"/>
           <SegmentURL media="bunny_2s_50kbit/bunny_2s7.m4s"/>
           <SegmentURL media="bunny_2s_50kbit/bunny_2s8.m4s"/>
```

Figura 6.17: Extracto de un ejemplo de fichero MPD

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

6.4.2. Arquitectura

La arquitectura básica de un sistema de distribución multimedia basado en la tecnología DASH está formada por un servidor HTTP y un cliente. Opcionalmente, se pueden añadir elementos intermedios, tales como cachés HTTP, proxies o pasarelas (sección 6.2.5), con el fin de mejorar la eficiencia del servicio.

6.4.2.1. Servidor

El elemento servidor almacena el fichero MPD con la descripción de los medios y los contenidos codificados siguiendo el estándar DASH. A diferencia de las arquitectu-

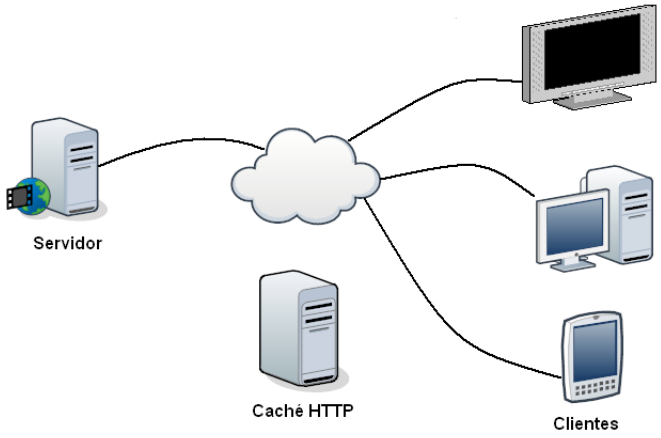


Figura 6.18: Arquitectura en streaming HTTP

ras *streaming* tradicionales, el servidor empleado para la distribución multimedia en la tecnología DASH es un servidor HTTP convencional.

6.4.2.2. Caché HTTP

Constituye un espacio de almacenamiento para peticiones HTTP y respuestas HTTP. Cuando se detecta una petición HTTP idéntica a alguna de las almacenadas, se devuelve la respuesta correspondiente previamente guardada, sin la necesidad de enviar esa petición al servidor final. Su uso se centra en la optimización, puesto que reducen el número de peticiones que debe atender el servidor.

6.4.2.3. Cliente

El cliente DASH será el encargado de solicitar los contenidos al servidor web mediante peticiones GET estándar y reproducirlos. Además, puede implementar algún tipo de lógica que permita llevar a cabo procesos de adaptación al ancho de banda disponible. A diferencia de otros sistemas *streaming* adaptativos, el control y la lógica de adaptación reside en el lado del cliente, por lo que los problemas relacionados con las cuestiones de escalabilidad en el lado del servidor quedan resueltos.

Sin embargo, la forma en la que los clientes llevan a cabo el proceso de estimación del ancho de banda y las decisiones referentes a la solicitud de los segmentos para llevar a cabo la adaptación de los contenidos están fuera del ámbito del estándar DASH. La especificación MPEG-DASH solo se refiere al fichero MPD y el formato de los segmentos. La forma en la que se realiza el envío del fichero MPD, la codificación de los medios y el comportamiento del cliente están fuera del ámbito del estándar.

6.4.3. Protocolos

Una de las principales ventajas de la tecnología DASH es el uso de protocolos ampliamente utilizados en el intercambio de información en Internet. Tradicionalmente, este tipo de protocolos no eran considerados viables para llevar a cabo la transmisión de contenido

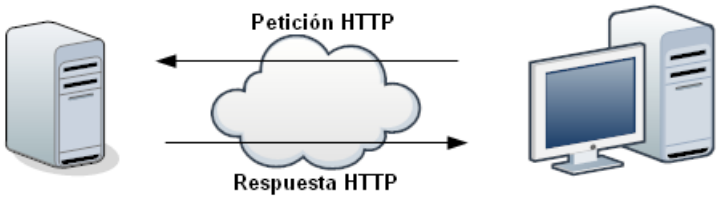


Figura 6.19: Protocolo HTTP

multimedia. Por ejemplo, los mecanismos de protección de errores y de retransmisión del protocolo de transporte TCP resultaban inviables para enviar información con requerimientos de tiempo real. Sin embargo, las mejoras introducidas con el estándar DASH permiten la utilización de este tipo de protocolos en escenarios *streaming*, dotando también de cierto grado de adaptación en función de las capacidades del cliente y del estado de la red.

DASH basa su funcionamiento en el conocido protocolo HTTP, cuyas principales características quedan resumidas a continuación.

6.4.3.1. HTTP

El protocolo HTTP (*Hypertext Transfer Protocol*) (RFC2616) es el protocolo empleado para el intercambio de información en Internet. HTTP está basado en el modelo de comunicación petición-respuesta entre un cliente y un servidor: el cliente realiza una solicitud HTTP, el servidor la procesa y envía la respuesta HTTP (Figura 6.19).

El protocolo HTTP se implementa sobre TCP, de manera que la comunicación es fiable y se garantiza la entrega de los mensajes al servidor y el envío al cliente. A diferencia de los protocolos vistos para *streaming*, HTTP es un protocolo sin estados, por lo que las peticiones se gestionan de forma independiente.

Cliente y servidor se intercambian 2 tipos de mensajes: peticiones (o solicitudes) y respuestas. Las peticiones son los mensajes que genera el cliente y las respuestas se corresponden con los mensajes cuyo origen es el servidor.

Petición HTTP

Las peticiones son mensajes que el cliente envía al servidor. Básicamente se componen de una línea de solicitud, una cabecera y un cuerpo de solicitud. La línea de solicitud indica el tipo para los mensajes de petición. La cabecera contiene los atributos del mensaje. Por último, el cuerpo del mensaje es opcional, en función del tipo de petición realizada.

La línea de solicitud está formada por el método, la URL de la petición y la versión del protocolo HTTP empleado (ver Figura 6.20). El método indica el tipo de petición realizada. Existen diferentes métodos:

- OPTIONS: permite a un cliente preguntar a un servidor por las opciones que soporta.

```
OPTIONS http://miservidor/index.html HTTP/1.1
```

Figura 6.20: Ejemplo de línea de solicitud para peticiones HTTP

```
HTTP/1.1 200 OK
HTTP/1.1 404 Not Found
HTTP/1.1 501 Method Not Implemented
```

Figura 6.21: Ejemplos de líneas de estado de respuestas HTTP

- HEAD: solicita un encabezado de un recurso que se encuentra en la URL que se indica junto con la petición.
- GET: solicita un recurso que se encuentra en la URL que se indica junto con la petición.
- POST: se emplea para solicitar un recurso a un servidor, enviando datos al programa que se encuentra en la URL especificada.
- PUT: envía datos a la URL especificada.
- DELETE: borra el recurso ubicado en la URL especificada.

Respuesta HTTP

La respuesta HTTP es el conjunto de información que fluye del servidor al cliente. Los mensajes de respuesta comienzan con una línea de estado. La línea de estado incluye la versión del protocolo HTTP empleado junto con un código de estado y una frase de razón (ver Figura 6.21). El código de estado es un campo numérico que indica el estado de la petición realizada por el cliente. La frase de razón es una descripción textual del código de estado que se envía.

Los códigos de estado se clasifican atendiendo a su valor en centenas. Distinguimos así diferentes tipos:

- tipo 200: indican el éxito de la operación
- tipo 300: son códigos de redirección y permiten al servidor indicar al cliente que el recurso solicitado se encuentra en una ubicación diferente.
- tipo 400: indican un error en la solicitud enviada por el cliente. Ejemplos que generan este tipo de respuesta son las peticiones de recursos que no existen en el servidor.
- tipo 500: indican un error en el servidor. Por ejemplo, cuando la versión del protocolo HTTP que emplea el cliente no es soportada por el servidor.

Capítulo 7

APLICACIONES MULTIMEDIA EN LA RED

En las redes IP (*Internet Protocol*), los paquetes se envían sin garantía de servicio y los recursos de red se comparten equitativamente. Es por eso que los problemas en cuanto a la calidad de servicio *streaming* en una red de datos se derivan principalmente de dos factores:

1. La red de datos está basada en conmutación de paquetes. Por lo tanto, la información no viaja siempre por el mismo camino, lo que provoca *jitter* o pérdida de paquetes.

2. Las aplicaciones multimedia tienen requerimientos de tiempo real. El retardo o latencia y la pérdida de paquetes puede llegar a ser muy perjudicial.

Evidentemente, el comportamiento de la aplicación multimedia durante el proceso de transmisión en la red depende en gran medida del protocolo de transporte de red empleado. Existen dos opciones, cada una de ellas presenta una serie de ventajas e inconvenientes frente a la otra y ya han sido esbozadas en capítulos previos: por un lado, el protocolo orientado a conexión TCP (*Transport Control Protocol*), y por otro lado, el protocolo no orientado a conexión UDP (*User Datagram Protocol*).

- TCP (RFC793): es el protocolo de transporte más empleado ya que ofrece un servicio de transferencia fiable de datos. Es un protocolo orientado a conexión, esto quiere decir que, antes de proceder a la transeferencia de los datos, los extremos que participan en el intercambio de datos negocian una conexión TCP mediante un proceso de establecimiento de conexión. Decimos que TCP es un protocolo fiable porque incorpora mecanismos de control de errores. Además, TCP implementa mecanismos de control de flujo, basados en técnicas de ventanas deslizantes, que evitan saturar el sistema receptor.

- UDP (RFC768): el protocolo UDP fue creado con el propósito de dar soporte a aplicaciones con requerimientos en tiempo real. Permite el envío de datos sin que se haya establecido previamente una conexión. Tampoco dispone de mecanismos de control de flujo ni corrección de errores, de manera que las aplicaciones que emplean UDP como protocolo de transporte asumen la posibilidad de que existan pérdidas.

En esta sección veremos la repercusión que causan las limitaciones tecnológicas y los factores derivados del funcionamiento de la red de conmutación de paquetes, como son el retardo, el *jitter* o la pérdida de paquetes y su impacto en la calidad de la sesión *streaming*.

7.1. Pérdidas

La pérdida de paquetes es un hecho común en todas las redes de conmutación de paquetes. Como ya se ha comentado, en este tipo de redes no hay ninguna reserva de recursos en un proceso previo al envío de la información. Como consecuencia, los paquetes se pueden perder. Puede ocurrir que, durante periodos de congestión, algún nodo de la red no sea capaz de procesar un paquete y lo descarte, dando lugar a una pérdida. A esto añadimos que el servicio de *streaming* tradicional a través de Internet emplea UDP como protocolo de la capa de transporte, el cual no ofrece ningún tipo de garantía en la entrega de los paquetes.

7.2. Retardo

Podemos definir el retardo como el tiempo que transcurre desde que se generan las muestras de audio/vídeo en el extremo emisor hasta que llegan al receptor. El retardo extremo a extremo tiene un efecto relevante en la percepción de la calidad.

Existen diversas contribuciones al retardo final extremo a extremo. Podemos diferenciar el retardo de procesamiento, de encolado, de transmisión y de propagación. La suma de todos ellos será el retardo final.

- Retardo de procesamiento: es el tiempo empleado en digitalizar la señal de audio y/o vídeo y, en algunos códecs, el tiempo que se añade como consecuencia de la compresión de la señal para reducir el ancho de banda consumido.
- Retardo de empaquetado: es el tiempo empleado en rellenar un paquete de datos con datos de audio y/o vídeo comprimidos.
- Retardo de serialización: es el tiempo requerido por un paquete para ser colocado en el enlace.
- Retardo de propagación: es el tiempo empleado para transportar la información de origen a destino a través del medio.
- Retardo de encolado: es el tiempo que un paquete espera para ser transmitido. Depende de los elementos que forman la red (*routers* etc.) y también de la cantidad de tráfico que viaje por la red.

7.3. Jitter

El *jitter* se define como la variación de retardo. Únicamente está presente en las redes de conmutación de datos, ya que cada paquete puede recorrer caminos distintos entre origen y destino y por tanto puede sufrir retardos distintos. Su efecto puede ser más perjudicial que el propio retardo. Si el paquete se retrasa más de lo debido, se dará por perdido, con la consiguiente disminución de la calidad.

En algunas aplicaciones el *jitter* se puede reducir mediante el almacenamiento temporal de los paquetes en el receptor a través de un *buffer*. Ese almacenamiento dura el tiempo

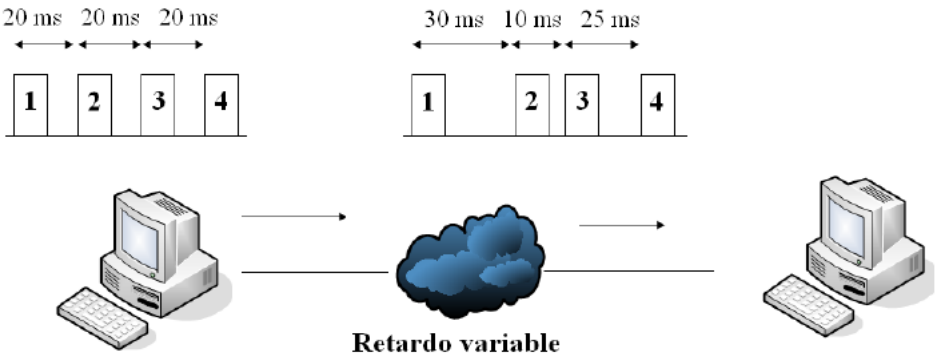


Figura 7.1: Ejemplo de jitter

suficiente para que los paquetes se puedan reordenar y reproducir en el orden correcto. El *buffer* de supresión del *jitter* presenta el inconveniente del aumento del retardo en el extremo receptor. Además, si la llegada de paquetes se produce en instantes en los que el *buffer* está lleno, los paquetes serán descartados, ocasionando una nueva fuente de pérdida de paquetes.

7.4. Ancho de banda

El ancho de banda se define como la cantidad de datos que pueden fluir a través de una conexión de red en un periodo de tiempo dado. Es uno de los factores más importantes en la ingeniería de redes. El *streaming* de audio y vídeo requiere grandes cantidades de ancho de banda y constituye uno de los puntos claves a la hora de mejorar la calidad percibida por el usuario. En esta línea, las aplicaciones de *streaming* adaptativo tienen como objetivo adaptar la calidad de los contenidos transmitidos al ancho de banda disponible con el fin de maximizar la experiencia de usuario.

7.5. Necesidad de QoS

En redes como Internet resulta muy complicado garantizar la QoS de servicios con requerimientos de tiempo real, ya que no tenemos control sobre el enrutamiento de los datos y pueden darse situaciones de elevada latencia y pérdida de paquetes durante la transmisión.

No debemos olvidar que, en el caso particular de los servicios *streaming*, las redes de conmutación de paquetes (como las redes IP) no fueron diseñadas con el propósito de transportar información multimedia. Aunque podamos transmitir contenidos de audio/vídeo y datos por la misma red, no va a ser posible tratarlos de la misma manera, ya que los primeros son mucho más sensibles al retardo, al *jitter* y a la pérdida de paquetes por tratarse de fuentes de tráfico de tiempo real y los segundos presentan exigencias más flexibles. Es por eso que se hace necesario el uso de mecanismos adicionales para complementar el servicio *best-effort* que ofrece IP con el fin de obtener una buena calidad de la sesión *streaming* (Wu et. al, 2001).

El concepto de calidad del vídeo puede ser visto desde dos perspectivas:

1. Desde el punto de vista del usuario: es decir, la percepción de la calidad del vídeo recibido en el cliente a la hora de su reproducción y visionado. Para este caso la degradación de la calidad se puede presentar de muchas maneras: por el efecto bloque, la pixelación, congelado de la imagen, etc.

   La medida de la calidad de los contenidos del vídeo recibido en el cliente puede realizarse mediante métricas objetivas o mediante métricas subjetivas. Las métricas objetivas están basadas en el uso de algortimos y las más empleadas son PSNR (*Peak Signal-to-Noise Ratio*) y SSIM (*Structural Similarity Index Metric*). Ambas requieren acceso a los contenidos originales para poder cuantificar la calidad de la imagen recibida. Por su parte, las métricas subjetivas se basan en las opiniones de los usuarios frente a los estímulos de vídeo recibidos. El estudio de la calidad de vídeo mediante evaluaciones subjetivas supone un consumo de tiempo elevado, por lo que, habitualmente, se recurre a métricas objetivas. A pesar del dominio de las métricas objetivas PSNR y SSIM, existen otras, como MOVIE (Seshadrinathan and Bovik, 2009), que obtienen un elevado grado de correlación con las métricas subjetivas. Su principal inconveniente, a día de hoy, es el elevadísimo coste computacional durante su cálculo.

2. Desde el punto de vista de la red: además de las métricas anteriores de calidad percibida, e independientemente del tipo de códec empleado, tenemos que tener en cuenta la calidad de la transmisión reflejada a través del nivel de servicio de la red (es decir, si la conexión es capaz de transportar el vídeo). Los parámetros de ancho de banda, retardo, *jitter* y pérdidas serán determinantes para decidir si la red está preparada para ofrecer tráfico en tiempo real.

Hoy en día aún existen una serie de limitaciones tecnológicas para ofertar un servicio de audio/vídeo de alta calidad a través de Internet. La dinámica de las redes tipo *best-effort* en términos de variaciones de ancho de banda y retardos hace que sea un problema proporcionar una buena calidad en las transmisiones *streaming*. Algunas técnicas de codificación implementan métodos de correción de errores (Sullivan and Wiegand, 2005). Otra opción que podemos plantear es mejorar la arquitectura de red: podemos sobredimensionar el ancho de banda del enlace y esperar que el retardo el *jitter* y las pérdidas no sean demasiado elevados (sin garantías). También podríamos solicitar la retransmisión de aquellos paquetes perdidos, pero aumentaríamos el retardo, empeorando aún más la calidad. Necesitamos, por tanto, otras soluciones para garantizar cierta QoS en los servicios de distribución de vídeo a través de Internet. Por ejemplo, algunos programas utilizan técnicas a nivel de aplicación, como son el empleo de *búffers* o algoritmos de codificación resistentes a pérdidas, buscando mitigar los efectos del retardo, el *jitter* y las pérdidas. Pero si el sistema final envía un número excesivo de paquetes sin tener en cuenta el ancho de banda disponible, la congestión es inevitable.

Por otro lado, los servicios con requerimientos temporales estrictos suelen transmitirse usando UDP. A diferencia de TCP (que puede modificar la tasa de envío para evitar la congestión), UDP carece del control de flujo. Por tanto es importante mejorar UDP con un mecanismo adicional de control de congestión para prevenir la pérdida de paquetes y

reducir el retardo.

Por su parte, TCP está adquieriendo cada vez más relevancia en la transmisión de contenido multimedia, en gran parte gracias a los estándares surgidos a tal efecto (sección 6.4). A pesar de que TCP cuenta con mecanismos de reenvío para evitar las pérdidas, son necesarios también los mecanismos de estimación de ancho de banda para poder adaptar la tasa de transmisión de los contenidos y evitar cortes en la reproducción debido a situaciones de congestión.

Parte III

Trabajos relacionados

Capítulo 8

ESTADO DEL ARTE

El desarrollo de la tecnología y al aumento del ancho de banda en la red de acceso permite la transmisión de contenidos multimedia en Internet. Este nuevo servicio hace posible, por ejemplo, la televisión a través de Internet o el vídeo a la carta, de manera que presenta un gran interés en diversos ámbitos. Los usuarios cada vez hacen un uso más intensivo de este tipo de servicios y los proveedores intentan ofrecer vídeos de mejor calidad para responder a esas necesidades.

Pero esta tecnología presenta dos problemas principalmente: el elevado consumo de recursos, tanto computacionales como de red, y la necesidad de unas condiciones de transmisión estables para poder garantizar una cierta calidad. Estos problemas derivan directamente de la ausencia del control de calidad en Internet, puesto que las redes basadas en una tecnología de envío *best-effort* no son la mejor opción para proporcionar un servicio de *streaming*. Estrechamente relacionado con este tema se han hecho muchas investigaciones para mejorar el nivel de operación de red. Por ejemplo, los *routers* utilizan un sistema para determinar el mejor *path* posible en base a las rutas disponibles en la tabla de rutas según la dirección de destino. En este caso, los mecanismos para evitar la congestión se basan en soluciones en los que la inteligencia está en la red (para este ejemplo, en los *routers*). Sin embargo esto no será suficiente para servicios en tiempo real como la tecnología *streaming*, en los que el sistema final será el que realiza un papel crucial respondiendo apropiadamente a estas señales de congestión.

Como consecuencia del interés que están generando estos servicios y los desafíos técnicos que presentan, han surgido numerosos trabajos en este campo que intentan buscar soluciones a las debilidades de esta tecnología. El objetivo final será mejorar la experiencia de usuario desarrollando tecnologías que subsanen estos problemas.

A continuación abordamos las dos cuestiones claves en las investigaciones relacionadas con los sistemas de distribución multimedia adaptativos: por un lado el cálculo del ancho de banda disponible en la red y por otro lado la forma en la que se lleva a cabo la adaptación de los contenidos a los recursos libres en la red. En la última sección del capítulo analizamos conjuntamente estas dos cuestiones, estudiando los trabajos relacionados con los sistemas *streaming* adaptativos.

8.1. Estimación del ancho de banda

En las redes de paquetes, los términos de ancho de banda y throughput caracterizan la cantidad de datos que la red puede transferir por unidad de tiempo. Las técnicas de estimación de ancho de banda (Prasad et. al, 2003) nos permitirán calcular la capacidad disponible que hay en un enlace.

La correcta estimación del ancho de banda disponible es interesante tanto para usuarios como para proveedores. Para los primeros, las técnicas de estimación facilitarán la optimización del comportamiento end-to-end de la transmisión. Para los segundos resulta provechoso en términos de ingeniería de tráfico y gestión de la capacidad.

El cálculo puede resultar trivial o relativamente sencillo si se dispone de acceso a los routers o switches conectados al enlace de interés: simplemente será necesario leer la información relativa al bitrate del enlace o los bytes transmitidos, con ayuda de un protocolo de administración de red, como SNMP (RFC1157). Sin embargo, los usuarios finales no tienen acceso a este tipo de recursos y deben estimar el ancho de banda del enlace sin la ayuda de la información obtenida de los equipos de red.

La solución se presenta, principalmente, en dos posibilidades: las técnicas de estimación de ancho de banda activas y las técnicas pasivas.

Técnicas activas

Las técnicas activas o intrusivas se basan en la inyección de carga adicional en la red, empleando ese tráfico extra para obtener datos (tales como los tiempos de ida y vuelta o medidas de dispersión de los paquetes) con los que estimar el ancho de banda disponible.

Hay muchas herramientas que emplean esta técnica: *pathChar (Downey, 1999)*, *Cprobe (Carter and Crovella, 1996)*, *iperf* o *pathLoad* (Jain and Dovrolis, 2002) son algunos ejemplos de ello. PathLoad emplea paquetes UDP para efectuar la estimación del ancho de banda y obtener un rango de valores correspondiente a la variación del ancho de banda que ocurre durante el proceso de medida. Por su parte, iperf es un ejemplo de herramienta que emplea largas transferencias de paquetes TCP para medir el throughput extremo a extremo, permitiendo además el manejo de múltiples transferencias paralelas.

El problema que presentan estas técnicas deriva principalmente de la influencia que ejercen en las condiciones de transmisión de la red. Dicho de otro modo, este tipo de técnicas pueden producir situaciones de congestión por el tráfico adicional que se introduce para hacer la estimación. Por ejemplo, si queremos realizar una adaptación a lo largo de una transmisión de contenidos de manera continua, necesitaremos que la estimación se haga de forma periódica. En este caso, la carga adicional necesaria para llevar a cabo esta técnica podría presentar un volumen de tráfico significativo, pudiendo incluso provocar situaciones de pérdidas de paquetes cuando los recursos de red estén altamente ocupados.

Técnicas pasivas

En las técnicas pasivas o no intrusivas, la estimación del ancho de banda se hace a partir de tráfico que ya existe en la red, sin necesidad de inyectar tráfico adicional. Las métricas empleadas son muy variadas: se puede utilizar las medidas de la pérdida de paquetes, el retardo o incluso el *buffer* del cliente, ya que puede verse afectado por situaciones anómalas de la red.

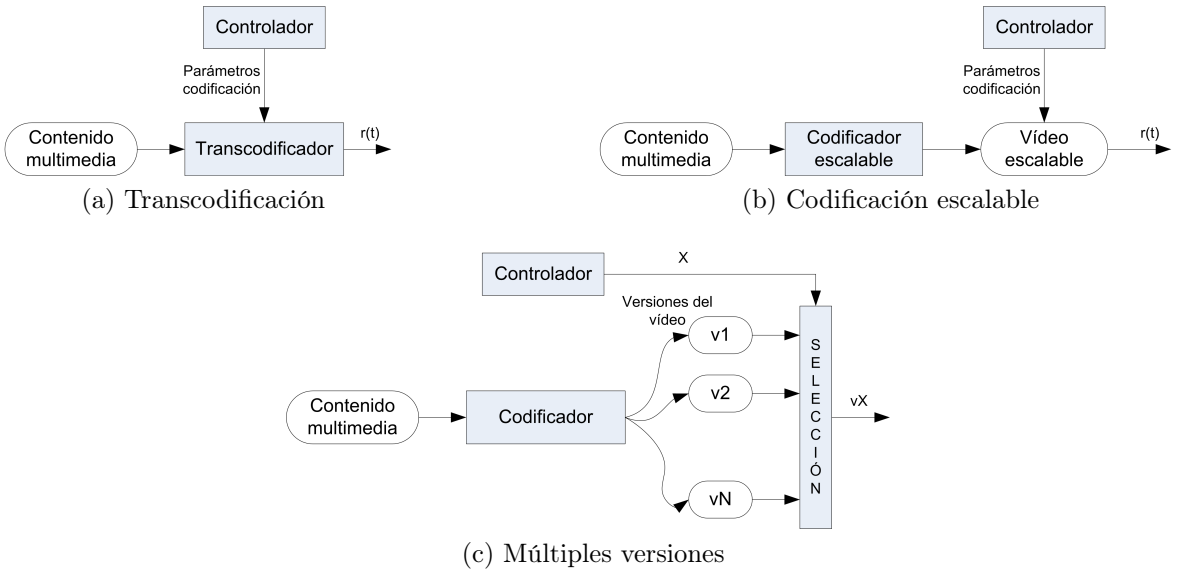
La pérdida de paquetes es un claro indicativo acerca del estado de la transmisión. Algunos autores emplean esta métrica para determinar si el ancho de banda en la red es suficiente o no para continuar con una transmisión fluida. Por ejemplo, el trabajo presentado por (Wanxiang and Zhenming, 2001) emplea información de la tasa de pérdidas que se incluye en los informes del protocolo RTCP. Por su parte, (Wu et. al, 2000) emplean el mismo mecanismo de estimación del ancho de banda disponible en la red, basándose únicamente en medidas del porcentaje de paquetes perdidos en el receptor.

El uso del porcentaje de pérdidas como indicativo de congestión es una métrica ampliamente utilizada y aceptada. Sin embargo, algunos autores (Bolot and Turletti (1998)) apuntan que la estimación de la congestión de red puede afinarse si, además, se emplea la métrica del *jitter*. A partir de esta propuesta, otros trabajos más recientes combinan ambas métricas para estimar el ancho de banda en la red. Por ejemplo, (Ahmad et. al, 2007) basan sus cálculos en el porcentaje de pérdidas y en el *jitter*. El uso de estas dos métricas permite a los autores diferenciar entre dos tipos de congestiones en la red: por un lado las congestiones persistentes y por otro lado las congestiones transitorias. La primera de ellas se detecta a partir de una estimación de la probabilidad de la tasa de paquetes perdidos. La segunda es detectada a partir de los valores del *jitter*. La principal ventaja de esta combinación de métricas para realizar la estimación del ancho de banda es la posibilidad de adelantarse a las situaciones de congestión severas, puesto que el *jitter* es un buen indicativo de congestión que se manifiesta antes de que se produzca un volumen de pérdidas en la red elevado (RFC3550). (Tionardi and Hartanto, 2003) también emplean el *jitter* para detectar situaciones de sobrecarga en la red, comparando los valores detectados con umbrales establecidos para tal fin.

En otros estudios emplean el orden de los números de secuencia recibidos en el cliente, de manera que los paquetes que se descartan o reordenan se emplean como indicador de congestión (Papadimitriou and Tsaoussidis, 2007).

Para algunas aplicaciones y escenarios, el *buffer* del cliente resulta un indicador válido acerca del estado de la transmisión, puesto que si los niveles de ocupación del *buffer* caen por debajo de unos valores establecidos puede ser un indicativo de congestión (Frojdh et. al, 2006). Además del *buffer* del cliente, otros autores añaden la monitorización del *buffer* de red para escenarios móviles inalámbricos (Baldo et. al, 2004). El principal problema, tal y como se destacó anteriormente, recae en el hecho de que no siempre es posible acceder a los parámetros de red de los dispositivos.

Para la temática abordada en esta tesis doctoral, el uso de técnicas no instrusivas para llevar a cabo los procesos de estimación de ancho de banda resulta especialmente interesante. Aprovechar la información que podemos extraer de la propia transmisión de



(a) Transcodificación

(b) Codificación escalable

(c) Múltiples versiones

Figura 8.1: Técnicas de adaptación

los contenidos es una clara ventaja para escenarios en los que los datos manejados tienen fuertes restricciones en cuanto a pérdidas y retardos. La inyección de tráfico adicional puede empeorar el resultado de la comunicación, por lo que, para estos escenarios, evitaremos esa opción. Los algoritmos de estimación diseñados (Fraga et. al, 2011; Pozueco et. al, 2013) y analizados en profundidad en la Parte IV «Aportaciones», suponen una sobrecarga en la red menor del 0.1 % en ambos casos, lo que permite calificar los sistemas como no intrusivos.

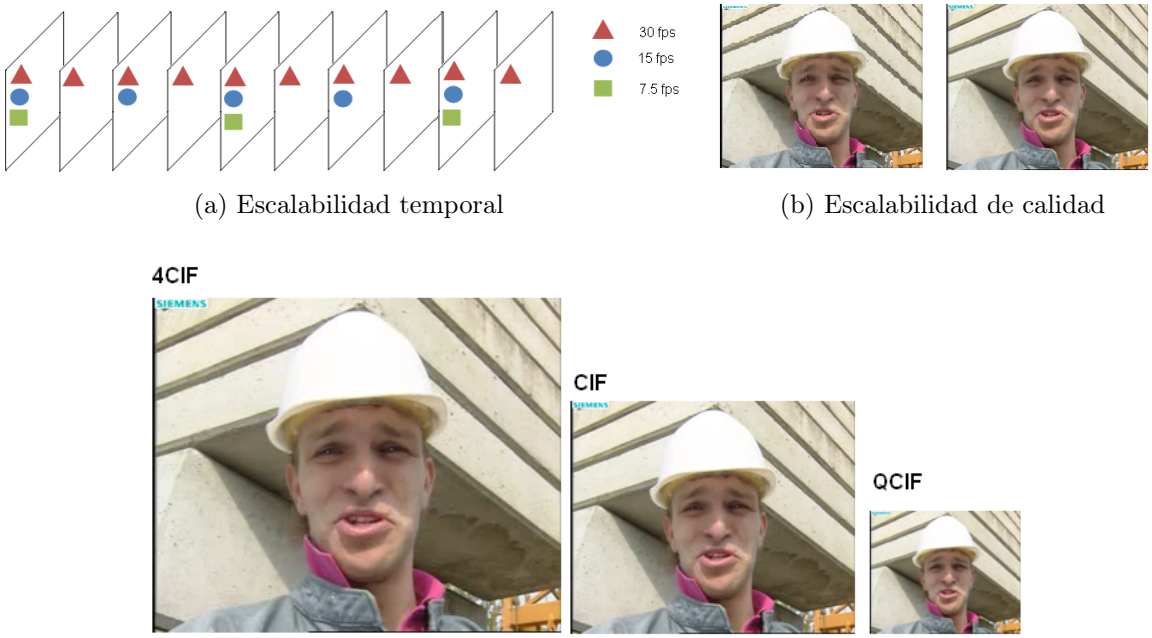
8.2. Adaptación del contenido

La siguiente cuestión que se plantea se refiere a la forma en la que se lleva a cabo la adaptación de los contenidos. Entre las diferentes posibilidades, se encuentran: la opción de disponer de múltiples versiones de diferentes características para el mismo contenido, las técnicas de *transcoding* y los formatos de codificación escalables (Figura 8.1). También existen otras técnicas más elaboradas en las que se analiza información semántica (donde se detectan eventos en el vídeo) o se estudian las relaciones de los elementos estructurales de las imágenes (identificando los frames representativos y eliminando aquéllos menos importantes o reduciendo la calidad en las zonas menos relevantes) (Chang and Vetro, 2005).

En cualquier caso, todas las opciones se pueden utilizar para adaptar la tasa de codificación, modificando características de los contenidos multimedia tales como la resolución (dimensiones del vídeo), el número de *frames* por segundo o la calidad de los *frames*. Ejemplos de cada uno de estos niveles de escalabilidad los vemos en la Figura 8.2

Multiplicidad de versiones

Las técnicas de multiplicidad de versiones requieren cierta capacidad de almacenamiento en la cabecera del servicio, ya que habrá que tener almacenadas diferentes copias



(a) Escalabilidad temporal

(b) Escalabilidad de calidad

(c) Escalabilidad espacial

Figura 8.2: Ejemplos de escalabilidad

del vídeo, cada una de ellas con unos niveles de calidad distintos. Un ejemplo de este tipo de técnicas lo presenta la tecnología DASH (Dynamic Adaptive Streaming over HTTP) (ISO/IEC, 2014), donde el servidor almacena diferentes versiones de los mismos contenidos pero con diferente bitrate.

Las principales opciones comerciales optan por este tipo de soporte para llevar a cabo ciertos procesos adaptativos. Por ejemplo, *Microsoft* proporciona el servicio *IIS Smooth Streaming* (Zambielli, 2009), donde las diferentes versiones del contenido se pueden codificar con *bitrates* y resoluciones configurables, siendo la opción por defecto la codificación de 7 versiones de los contenidos, en el rago de tasas entre los 300 kbps y 2.4 Mbps.

Por su parte, *Adobe* también ha desarrollado un servicio web para realizar *streaming* adaptativo, en el que el servidor almacena diferentes calidades y resoluciones de los contenidos y conmuta entre ellos durante la reproducción con el fin de adaptarse al ancho de banda y la CPU del cliente. El servicio está basado en un protocolo propietario, denominado RTMP (Parmar and Thornburgh, 2012).

Más recientemente, *Apple* se ha unido al desarrollo de soluciones *streaming* adaptativas con HLS (*HTTP Adaptive Live Streaming*) (Pantos and May, 2010). Al igual que sus competidores, el servidor almacena varias versiones de los contenidos y facilita al cliente dicha información para que éste decida qué contenidos descargar.

La sencillez a la hora de llevar a cabo el proceso de adaptación es la ventaja más destacada de esta opción. Únicamente requiere disponer de una correcta descripción de los medios almacenados, de forma que el cliente o el servidor puedan seleccionar la opción más adecuada para cada circunstacia. Pero la generación de las diferentes versiones del

contenido requiere una preparación previa de los mismos. Además, deberemos añadir los costes asociados al almacenamiento al balance de ventajas e inconvenientes.

Transcodificación

La opción de transcodificar exige cierto coste computacional, ya que es necesario decodificar el contenido y volverlo a codificar de acuerdo a las restricciones del usuario final (Xin et. al, 2005). Con las técnicas de transcoding es posible convertir un formato de vídeo en otro distinto, variando las características antes mencionadas, como el *bit rate*, el *frame rate* o la resolución espacial. Las primeras aplicaciones de las técnicas de transcoding tienen que ver con la modificación del *bit rate* para acomodar el vídeo al ancho de banda del canal.

La transcodificación se presenta como una opción controvertida, en la que es necesario hacer balance acerca de los beneficios en la mejora de la calidad percibida frente al coste para llevarlo a cabo, especialemente en escenarios móviles. Algunos autores afirman que, en algunos casos, las variaciones en el *bit rate* apenas son perceptibles por el ser humano, de forma que estas técnicas presentan un derroche de recursos (Ma et. al, 2011). Sin embargo, (Shen et. al, 2008) suplen la problemática del coste computacional mediante un diseño inteligente que reutiliza información (el modo de codificación o información de movimiento) en el núcleo de la red, sosteniendo que las técnicas de transcodificación son la mejor opción para adaptar el contenido en función de la potencia de los terminales.

La versatilidad es, quizá, el mejor atributo que puede presentar esta opción de adaptación. Mediante la transcodificación, el sistema encargado de llevar a cabo el proceso de adaptación dispone de un rango continuo de valores entre los que seleccionar la tasa de codificación. En otros sistemas, donde los contenidos se generan previamente, únicamente existe un conjunto discreto de valores entre los que elegir el *bitrate* más adecuado. Por tanto, el aprovechamiento del ancho de banda disponible puede ser significativamente mejor empleando técnicas de transcodificación.

Codificadores escalables

Una tercera opción nos permite tener diferentes niveles de escalabilidad en un único *stream* de vídeo. De esta manera, no será necesario contar con varias versiones de los mismos contenidos con diferentes niveles de calidad, consiguiendo así un importante ahorro en el espacio de almacenamiento. Esta idea además permite servir a un amplio rango de terminales, sobre redes heterogéneas, con una única versión del vídeo que contará con varias capas de calidad. En función de las características del estado de la red del usuario, se accederán a las capas que conformen el *stream* con mayor o menor calidad, en función de las necesidades. Esta tecnología recibe el nombre de SVC (*Scalable Video Coding*) (Schwarz et. al, 2007), y proporciona escalabilidad en la codificación, transmisión y decodificación, pudiendo realizar la adaptación a un coste computacional bajo.

La estandarización de SVC finalizó en el año 2007 como una extensión del estándar H.264/AVC. Desde entonces, diversos trabajos se centran en aspectos relacionados con la eficiencia de codificación y la transmisión. Wenger et al. han sido uno de los grupos más

activos en este último aspecto, discutiendo ampliamente acerca de las especificaciones de la carga útil de RTP para SVC (Wenger et. al, 2006) y el transporte y la señalización de SVC en las redes IP (Wenger et. al, 2007). Además, otros autores como (Seo et. al, 2010) realizan propuestas para llevar a cabo un empaquetado eficiente de los contenidos codificados en SVC en paquetes RTP para su transmisión, de forma que el retardo introducido y la carga computacional se ven reducidas gracias al sistema propuesto, basado en información relativa a la prioridad de las capas de SVC.

Obviamente, el uso de SVC conjuntamente con algoritmos para realizar la adaptación de los contenidos no se haría esperar, y los autores (Chen et. al, 2007) realizan una de las primeras propuestas para este tipo de arquitecturas. Se trata de un prototipo sencillo, que únicamente explota la escalabilidad temporal que brinda SVC. Otros trabajos incorporarán la escalabilidad de calidad y la escalabilidad espacial en sus diseños (Peng et. al, 2008), teniendo en cuenta la dependencia existente entre las capas a la hora de extraer un bitstream válido y con la mayor calidad posible para las condiciones dadas.

Pero no todos los trabajos se centran en el ámbito del transporte del contenido y la adaptación. Otro aspecto importante es el relacionado con la medida de la calidad mediante métricas objetivas. Los autores (Sohn et. al, 2010) desarrollan una métrica, compatible con SVC, que permite medir la calidad del *bitstream*. Proponen un tipo de métrica full-reference, que presenta un elevado grado de correlación con los resultados subjetivos obtenidos para las mismas secuencias de vídeo.

Como vemos, la estandarización de este nuevo codificador supone un foco de nuevos trabajos de investigación en diversos ámbitos: desde cuestiones técnicas hasta aplicaciones prácticas en diversos escenarios.

SVC presenta la ventaja de la escalabilidad a un coste computacional abordable si a nuestro servicio accede un gran número de usuarios, ya que los requerimientos de computación para adaptar los formatos son mucho menores. Esto es posible gracias a que este sistema permite una adaptación eficiente directamente a nivel de *bitstream*, sin la necesidad de transcodificar, pudiendo extraer las capas con operaciones de baja complejidad, de manera que la adaptación a las condiciones de red y de los terminales es relativamente sencilla.

Al tratarse de un estándar de codificación relativamente reciente, la adopción por parte de la industria puede ser lenta, aunque ya existen empresas de soluciones para videoconferencia que implementan SVC en su arquitectura[1].

8.3. Estimación de ancho de banda y adaptación de contenidos

Las notorias dificultades para la transmisión de datos multimedia en Internet, debido a variaciones en el *throughput*, los retardos y las pérdidas, han marcado el desarrollo de numerosos trabajos de investigación. Algunos estudios proponen dotar de cierta QoS a las

[1] http://www.vidyo.com/

redes *best-effort* con mecanismos de reserva de red a la vez que se adapte el comportamiento de la aplicación a los servicios que proporciona la red y a las características variables con el tiempo de los canales sobre los cuales se envían los paquetes de datos (Bolot and Turletti, 1998). Sin embargo, los esfuerzos se encaminan hacia soluciones basadas en el control de la transmisión *end-to-end*. Uniendo los conceptos previamente introducidos, de estimación de la congestión y adaptación de los contenidos, podemos mejorar la calidad de un servicio *streaming*.

Este concepto ha sido desarrollado, entre otros, por el trabajo presentado por (Arsan, 2008) y, posteriormente, completado en (Arsan, 2012). En él, la arquitectura propuesta incluye un controlador de ancho de banda, un controlador de la calidad de servicio y un centro de decisión. Estos tres componentes realizan las tareas de estimación y adaptación. La principal diferencia de este trabajo frente a otros es el empleo de dos sistemas de estimación (correspondientes con los dos primeros citados elementos de la arquitectura). Por un lado, desarrollan un estimador activo basado en la herramienta *pathChirp*. Pero además, implementan un segundo sistema de estimación, esta vez pasivo, basado en los valores de retardo y paquetes perdidos de los paquetes RTCP. La información que arrojan los dos módulos de estimación se emplea en el tercer módulo de decisión. Su objetivo es hacer que la calidad en el lado del receptor sea razonable para las condiciones de red disponibles.

El empleo de técnicas activas de estimación del ancho de banda no es común en los escenarios de vídeo *streaming*. Aún así, algunos trabajos se basan únicamente en técnicas intrusivas para estimar el ancho de banda. Por ejemplo, el mecanismo elegido por (Rejaie et. al, 1999) consiste en la inyección de paquetes de prueba para medir los cuellos de botella. Pese a que existen algunas propuestas, la tónica general consiste en utilizar la información extraída del propio protocolo RTCP, no necesitando información adicional.

Explotar las ventajas que nos ofrecen los protocolos es el objetivo principal de muchos trabajos, como el desarrollado por (Bouras et. al, 2008) o (Ling and ShaoWen, 2009). En el primero de ellos, la información de congestión se emplea para posteriormente controlar ta tasa de codificación. Los autores (Ling and ShaoWen, 2009) incluyen además mecanismos para suavizar las medidas recogidas con el fin de evitar información sesgada de medidas que se basan únicamente en la tasa de paquetes perdidos, evitando oscilaciones durante el ajuste de la tasa transmitida.

Muchos de los trabajos iniciales en el campo de la estimación y adaptación basan el control de flujo en las medidas de la tasa de pérdidas de paquetes. Es el caso del sistema presentado por (Wanxiang and Zhenming, 2001), donde la información de los paquetes perdidos se extrae directamente de los informes RTCP del cliente. Si el porcentaje de paquetes perdidos supera un umbral, la tasa de transmisión se decrementa. Si está por debajo del umbral, se incrementa. Los incrementos que se aplican a la tasa de transmisión son aditivos y los decrementos multiplicativos. En este caso no implementan el sistema real, si no que construyen un modelo con la herramienta NS (*Network Simulator*)[2].

La forma en la que se llevan a cabo los incrementos y decrementos en la tasa de transmisión es clave en los esquemas de adaptación y existen estudios que analizan las

[2]http://www.isi.edu/nsnam/ns/

consecuencias derivadas de aplicar unos algoritmos u otros. Por ejemplo, (Sisalem and Wolisz, 1999), concluyen que los algoritmos que emplean incrementos multiplicativos sufren mayor pérdida de paquetes. Además, los incrementos o decrementos multiplicativos provocan oscilaciones de tasa a la hora de adaptar los contenidos (Rejaie et. al, 1999).

Detectar la congestión en la red a través de la pérdida de paquetes también ha sido estudiado por (Wu et. al, 2000). Su investigación constituye un ejemplo de control de tasa en el origen, donde el emisor es el responsable de adaptar la tasa de transmisión del vídeo. El *feedback* recibido del cliente permite al servidor implementar el mecanismo de adaptación. El origen sondea el ancho de banda disponible ajustando la tasa de transmisión de forma que el ratio de paquetes perdidos permanezca debajo de un umbral. El ajuste se basa en incrementos aditivos y decrementos multiplicativos.

La pérdida de paquetes es un parámetro bastante fiable para la detección de congestiones en la red. Sin embargo, no da ninguna información útil sobre el ancho de banda disponible para efectuar un incremento de la tasa. Por tanto, otros autores utilizan este parámetro en combinación con otros indicadores. Por ejemplo, (Ahmad et. al, 2007) proponen un mecanismo de control de congestión basado en RTP (*Real-time Transport Protocol*), donde usan el *jitter* para detectar congestiones transitorias (anticipándose a la congestión y regulando dinámicamente la tasa de transmisión de acuerdo a las condiciones de red antes de que las pérdidas ocurran) y los paquetes perdidos para las congestiones persistentes. El valor de las métricas analizadas permite clasificar el nivel de congestión mediante umbrales. A continuación, un algoritmo de adaptación decrementa la tasa de paquetes enviados o la aumenta con incrementos aditivos.

Otras métricas novedosas las encontramos en el trabajo de (Papadimitriou and Tsaoussidis, 2007), donde se propone un nuevo protocolo para el envío adaptativo de vídeo sobre Internet. El receptor usa los paquetes que se descartan o reordenan como un indicador de congestión. De esta forma el mecanismo de congestión se lanza cuando un paquete se recibe llevando un número de secuencia mayor que el número de secuencia esperado o cuando el receptor no adquiere ningún paquete dentro de un intervalo de *timeout*.

Como vemos, la mayoría de las investigaciones sigue una estructura similar, donde el cliente procesa la información recibida, mide ciertos parámetros de control y envía esta información de retroalimentación de vuelta al servidor. Después, un algoritmo estima el ancho de banda disponible y el servidor realiza la adaptación de los contenidos a ese valor.

Otras implementaciones, sobre todo las basadas en HTTP *streaming*, modifican ligeramente los roles anteriores, de forma que el módulo de estimación del ancho de banda se encuentra implementado en el cliente, quedando el servidor relegado únicamente al envío de los contenidos solicitados. Sin embargo, algunos autores optan por mantener un esquema clásico de sistema adaptativo, donde la lógica de adaptación reside en el servidor. Los autores (De Cicco et. al, 2011) implementan un sistema de estas características, empleando el estándar de streaming HTTP sobre TCP con la peculiaridad de concentrar los algoritmos de estimación y adaptación en el servidor, empleando información de *feedback*, y disponiendo de múltiples versiones de los contenidos en el servidor para llevar a cabo la adaptación.

Pero además de una arquitectura clásica basada en cliente-servidor, también encontramos trabajos en los que los procesos de adaptación son llevados a cabo por elementos intermedios en la red. Este tipo de configuraciones son frecuentes en sistemas basados en codificadores escalables, puesto que los elementos intermedios únicamente contraen la tarea de añadir o eliminar capas en el *stream* de vídeo que retransmiten (Grafl et. al, 2013). Sin embargo, (Shah and Nawaz, 2014) acomodan esta idea para un sistema basado en transcodificación, donde un elemento intermedio en la red actúa como un proxy multimedia transparente y proporciona la transcodificación de los contenidos multimedia entre el servidor y los clientes.

Si nos vamos al campo de la adaptación empleando la tecnología SVC, las tareas de adaptación se simplifican notablemente. Existen diversas propuestas,tanto teóricas como prácticas, para los sistemas basados en codificadores escalables. Por ejemplo, (Kofler et. al, 2008) desarrollan un *proxy* RTP/RTCP en un *router* wifi, empleando la información RTCP de *feedback* para los propósitos de adaptación en los elementos de red, pero dejando para trabajos futuros la implementación de una aplicación de control que determine los parámetros óptimos de adaptación. A pesar de constituir una primera aproximación en el campo de la adaptación con SVC, el trabajo presentado por Kofler et al. aborda una temática relevante y es, las sesiones concurrentes que el sistema puede manejar. Este trabajo se continúa en (Kofler et. al, 2011), donde los paquetes de *feedback* RTCP se reemplazan por la monitorizacion del retardo en las colas del *router* para detectar la degradación en el *throughput* de la red. Con ello consiguen una respuesta mejorada con respecto a escenarios basados en control de la congestión *end-to-end* que emplean información de *feedback*. Sin embargo, la evaluación se lleva a cabo teniendo en cuenta únicamente la escalabilidad espacial y, el acceso y control de los dispositivos de encaminamiento en la red no siempre es posible.

En la literatura encontramos abundantes referencias al empleo de elementos intermedios que realizan las tareas de adaptación. La técnica más empleada en este tipo de adaptaciones consiste en extraer capas del *bitstream* SVC. Por ejemplo, los autores en (Wenger et. al, 2007; Kuschnig et. al, 2008) proponen y analizan diferentes diseños e implementaciones de adaptación RTP/RTSP, para contenidos SVC, basada en MANEs (*Media Aware Nerwork Elements*). El primero de los trabajos se centra en un análisis teórico acerca del estándar y las topologías relevantes, tanto para escenarios multicast como para escenarios unicast. La variedad de terminales con diferentes características que acceden al servicio de vídeo *streaming* pueden verse favorecidos con el empleo de los elementos intermedios que acomodan el *stream* transmitido para cada sesión. El segundo de los trabajos mencionados (Kuschnig et. al (2008)) analiza también las técnicas de adaptación empleando MANEs, pero también da un paso más y estudia otros mecanismos de adaptación basados en la descripción mediante metadatos. Para esta última opción, la especificación MPEG-21 *Digital Item Adaptation* (DIA) (Timmerer et. al, 2007) proporciona una serie de normativas, siendo la estudiada la denominada *generic Bitstream Syntax Descriptions* (gBSD). En ella, un documento XML describe la estructura del *bitstream* multimedia a alto nivel (capas, cabeceras o paquetes), de forma que la adaptación es independiente del codificador. Los autores concluyen que esta opción es una alternativa viable para SVC, a pesar de que muestra un comportamiento peor que un sistema adaptativo basado en

MANEs en términos de retardo *end-to-end* introducido.

A partir del concepto de adaptación empleando metadatos, (Choi et. al, 2007), proponen un sistema de adaptación dinámico basado en la capacidad del cliente y el *path* de red. Dicha información está contenida en los mensajes MPEG-21 *Digital Item Adaptation* (DIA) que envía el cliente. En el servidor, esa información es empleada por el *Adaption Decision Taking Engine* (ADTE), módulo encargado de realizar las decisiones de adaptación. El ADTE determina la combinación óptima de los niveles de calidad, los niveles temporales y espaciales y, posteriormente, un tercer módulo, denominado *Dynamic Extractor,* examina la cabecera de las unidades NAL para eliminar las capas del *bitstream* que no pertenencen a la combinación seleccionada.

La información que se puede incluir en los metadatos es muy variada. En el caso del trabajo presentado por (Shao et. al, 2010), la información de *feedback* que se envía es referente a la calidad percibida, las condiciones de red y las preferencias de usuario. El uso de este tipo de técnicas de adaptación constituye un buen número de trabajos relacionados en todos los ámbitos, incluyendo el desarrollo de servidores de transmisión que soporten SVC como parte de un framework MPEG-21 para un escenario de red heterogéneo, (Eberhard et. al, 2008).

Más recientemente, (Gorkemli and Tekalp, 2012) proporcionan una serie de guías para llevar a cabo la adaptación de streams SVC sobre protocolos como TCP o DCCP (*Datagram Congestion Control Protocol*), que ya implementan mecanismos de control de congestión. Recomiendan, entre otras medidas, retransmitir los paquetes perdidos de la capa base de acuerdo al estado del *buffer* de decodificación.

También podemos encontrar propuestas que unen la técnica SVC con la transmision sobre sistemas MIMO (*Multiple Input Multiple Output*) para mejorar la calidad visual en las aplicaciones de video inalámbricas (Yang et. al, 2012). Como vemos, la aplicación práctica de los esquemas de adaptación con codificadores escalables abarcan un amplio espectro de posibilidades.

Siguiendo con la idea de estimación de ancho de banda y adaptación empleando la tecnología SVC, (Kantarci, 2008) propone una arquitectura típica de cliente-servidor, basada en protocolos clásicos para la transmisión *streaming*, con RTP y RTCP. La información de *feedback* intercambiada se refiere al nivel de ocupación del *buffer* del cliente. A partir de dicha información, junto con el porcentaje de pérdidas, se categoriza el estado de congestión en la red.

Por su parte, (Nguyen and Ostermann, 2007) proponen un sistema *streaming* de tiempo real usando SVC. Emplean un algoritmo de control de la congestión basado en algoritmos *TCP-Friendly,* que tiene la cualidad de no requerir paquetes de prueba en la red para estimar el ancho de banda. Además, también emplean información de *feedback* del cliente relativa a su estado del *buffer*. En este caso, además de adaptar la tasa binaria a las variaciones de ancho de banda, optimizan la calidad visual en el cliente evitando cambios frecuentes en las resoluciones espacial o temporal.

En este sentido, (Bianchi et. al, 2009), también tienen en cuenta la calidad final percibida, priorizando los paquetes en función de su importancia para la calidad del vídeo, de manera que los paquetes que se descartan son los que tienen un impacto menos negativo. La propuesta para llevar a cabo este cometido se basa en un sistema de gestión que opera a nivel de red y que dispone de colas separadas para cada nivel de prioridad. De manera similar, los autores (Schierl et. al, 2010) utilizan algoritmos que priorizan los datos de acuerdo a su importancia y analizan la solución que proponen para SVC tanto en RTP como en HTTP *streaming*.

Como vemos, existe una amplia variedad de trabajos relacionados en el campo de la adaptación de vídeo *streaming* sobre UDP debido, principalmente, a su baja latencia. Sin embargo, a pesar de la tendencia general de emplear UDP como protocolo de envío de audio y vídeo en tiempo real, el *streaming* HTTP está alcanzando importantes cotas de interés. Tradicionalmente, TCP no era considerado un protocolo factible para el envío de audio/vídeo. Sin embargo, el *streaming* HTTP dispone de una serie de ventajas que lo posicionan como una tecnología popular en despliegues comerciales.

A pesar de las ventajas demostradas del uso de sistemas adaptativos frente a los no adaptativos en términos de calidad de experiencia del usuario (Alvarez et. al, 2013), algunas soluciones comerciales no implementan procesos de estimación de recursos y adaptación de los contenidos durante la transmisión de los mismos. Los autores (Huang et. al, 2012) llevan a cabo un estudio del comportamiento de diferentes servidores *streaming* comerciales, revelando que son incapaces de estimar el ancho de banda disponible y, por consiguiente, no pueden servir los vídeos con la máxima calidad posible. En cuanto a los cliente comerciales, (Akhshabi et. al, 2012) proporcionan una evaluación experimental en la que identifican las ineficiencias de los reproductores que analizan y comparan su comportamiento. Sin embargo, las diferentes opciones comerciales para el *streaming* HTTP (*Apple's HTTP Live Streaming, Adobe's Dynamic HTTP Streaming, y Microsoft's Smooth Streaming*) usan diferentes formatos en los segmentos y necesitan cliente propietarios. Para solucionar estos problemas surge el estándar DASH, permitiendo interoperabilidad entre los servidores y clientes de diferentes fabricantes (Michalos et. al, 2012).

En esta nueva tendencia del *streaming* mediante DASH, la adaptación de los contenidos se implementa creando múltiples alternativas (o versiones) y desplazando la lógica de adaptación al cliente. Será éste quien decida cuál será el siguiente segmento que se solicitará al servidor. La problemática relacionada con la generación de las múltiples versiones de los contenidos en el servidor es abordada por el trabajo propuesto por (Ma et. al, 2014). En él, se plantea un algoritmo dinámico que planifica tareas de transcodificación para generar las diferentes versiones de los contenidos a partir del contenido original, todo ello en un entorno distribuido en la nube, puesto que las tareas de transcodificación son especialmente costosas computacionalmente.

Debido a que la especificación del estándar DASH no tiene en cuenta el diseño del proceso de estimación y adaptación, la mayoría de los trabajos de invesitagación se centran en este aspecto. Los autores (Thang et. al, 2012) proponen un sistema de estimación del ancho de banda basado en el *throughput* de segmentos previos recibidos. En cuando al mecanismo de adaptación, emplean metadatos que se añaden a la sintaxis de DASH.

El matiz más cuestionable de esta propuesta es la negativa de los autores frente al uso de métricas basadas en niveles de ocupación del *buffer* del cliente. (Liu et. al, 2011) también se basan únicamente en la medida suavizada del *throughput* para detectar variaciones en el ancho de banda. El algoritmo adaptativo que proponen para MPEG-DASH lleva a cabo incrementos paso a paso y decrementos agresivos en el *bitrate* transmitido y los resultados de la evaluación muestran una fluctuación significativa en la calidad del vídeo.

En las arquitecturas basadas en protocolos de transporte fiables y orientados a conexión, como TCP, los propios mecanismos de control de flujo que incorporan pueden acarrear interrupciones durante la reproducción de los contenidos si el ancho de banda se reduce de manera significativa. En este sentido, (Biernacki and Tutschku, 2014) evalúan la influencia de las discapacidades de la red y su impacto en la calidad de vídeo, concluyendo que las implementaciones de estrategias de análisis y control del *buffer* en el lado del cliente ayudan a mejorar la calidad percibida por el usuario.

Controlar los niveles de ocupación del *buffer* del cliente permite garantizar, en cierta medida, una reproducción continuada de los contenidos. Conscientes de ello, los autores (De Cicco and Mascolo, 2014) implementan un módulo diseñado para tal fin. El controlador propuesto consta de varios estados en función de la cantidad de información disponible en el *buffer*. Dicha información se emplea posteriormente para seleccionar el nivel del vídeo que se envía del servidor al cliente.

La selección del *bitrate* que el cliente solicita al servidor debe tener en cuenta dos aspectos. Por un lado, si el *bitrate* es mucho menor que el ancho de banda disponible, el reproductor almacena contenidos a una velocidad superior a la que los consume, llegando a provocar situaciones denominadas *buffer-overflow*. Por el contrario, si el *bitrate* de los contenidos es notablemente superior al ancho de banda disponible, el nivel de ocupación del *buffer* puede llegar a cero (*buffer-underflow*). Evitar las situaciones de *buffer-underflow* y *buffer-overflow* es el objetivo principal del trabajo presentado por (Zhou et. al, 2012) y para ello establecen umbrales en el *buffer* del cliente.

Sin olvidar los códecs escalables, algunos trabajos proponen la integración de SVC con la transmisión HTTP. (Schierl et. al, 2010) muestran los beneficios de usar streaming HTTP adaptativo en combinación con SVC para evitar las interrupciones en el enlace en redes móviles. Sin embargo, otros autores como (Kofler et. al, 2012) evalúan la integración de DASH con *streams* SVC y concluyen que la tecnología no es factible para *streams* por debajo de 1Mbps.

Una de las ventajas más alabadas de DASH es su integración con los elementos de la infraestructura web actual: proxies, cachés y CDNs. La optimización de los algoritmos de adaptación en entornos de redes de distribución de contenidos es otro frente abierto por esta tecnología y está siendo activamente estudiado (Liu et. al, 2012).

Para finalizar este estudio de los trabajos anteriores, y aunque no es un objetivo de esta tesis doctoral, haremos una breve mención a los trabajos de análisis de la calidad de la experiencia (QoE) mediante evaluaciones subjetivas. El análisis subjetivo de sistemas *streaming* adaptativos es un proceso clave en la optimización de algoritmos de adaptación

basados en medidas de calidad de la experiencia. Este tipo de estudios también resultan claves a la hora de analizar la calidad percibida para nuevos codificadores, nuevos sistemas de distribución o, en general, para evaluar los efectos de las degradaciones en el vídeo. Por ejemplo, la evaluación subjetiva de los codificadores escalables fue analizada por (Oelbaum et. al, 2008). En este trabajo presentan un estudio subjetivo formado por 20 usuarios y 12 secuencias de vídeo diferentes. Los resultados muestran que el códec H.264/SVC ofrece una calidad comparable con el códec de H.264/AVC (*Advanced Video Coding*) con un ligero incremento de la sobrecarga y la complejidad del decodificador.

Otros estudios llevados a cabo por (Mok et. al, 2012) se centran en la tecnología DASH, con la realización de test subjetivos a 24 participantes para evaluar 11 secuencias diferentes. Las conclusiones de este estudio muestran que los usuarios prefieren cambios graduales en la calidad de los contenidos frente a los cambios abruptos. Sin embargo, la construcción de estrategias óptimas de adaptación basadas en métricas de calidad de experiencia para sistemas de vídeo *streaming* aún presenta interrogantes en lo referente a las preferencias de usuario en los cambios de calidad (Cranley et. al, 2006). Cuándo y en qué situaciones un sistema adaptativo mejora la calidad percibida por el usuario son todavía cuestiones controvertidas y plantean dos tendencias opuestas en el campo de los servicios de vídeo en entornos *best-effort*: los sistemas adaptativos frente a los no adaptativos.

Las investigaciones de los trabajos anteriores sirven como punto de partida al estudio, diseño, implementación y análisis de diferentes sistemas de adaptación para la tecnología *streaming*. En una primera incursión en el campo de los sistemas adaptativos, proponemos un sistema basado en transcodificación (Fraga et. al, 2011), donde la estimación del ancho de banda se realiza a partir de información observada acerca de la transmisión del propio contenido multimedia. A pesar de obtener resultados de adaptación estables, el principal problema se relaciona con el elevado consumo de recursos en el servidor. Los resultados obtenidos nos conducen a mejorar el sistema de estimación y adaptación con una segunda propuesta basada en codificadores escalables (Pozueco et. al, 2013), mejorando los puntos débiles de la primera aproximación. Ambos sistemas emplean métricas de estimación de ancho de banda clásicas, como el *jitter* o los paquetes perdidos, pero además incluímos una métrica novedosa basada en la linealidad de los tiempos de recepción de los paquetes RTP en el cliente. Finalmente, para solucionar los problemas relativos a la escalabilidad del sistema y facilitar el despliegue en la infraestructura de red actual, proponemos un tercer sistema basado en HTTP streaming (Pozueco et. al, 2014). Los siguientes capítulos profundizan sobre los tres sistemas adaptativos diseñados.

Parte IV

Aportaciones

Capítulo 9

CONSIDERACIONES PREVIAS

En los sucesivos capítulos se describe el proceso de diseño, implementación y evaluación de algoritmos para sistemas de vídeo adaptativo. Para realizar estas tareas se llevaron a cabo una serie de experimentos de laboratorio. El mayor problema al que nos enfrentamos es abordar, de manera eficiente, el elevado número de pruebas que es necesario hacer, puesto que el tiempo y los recursos para realizar este tipo de ensayos suele ser relativamente elevado: hay que configurar correctamente los equipos y aplicaciones, es necesario diseñar los experimentos que queremos realizar, configurar los escenarios en los que se va a desarrollar cada uno de los experimentos y recoger todos los datos necesarios para analizar posteriormente.

Por tanto, resultará interesante el empleo de algún método o proceso que facilite estas tareas y permita definir los entornos de pruebas, los pasos a seguir a la hora de realizar dichos experimentos y el análisis de los resultados, consiguiendo así automatizar los procesos de manera sencilla.

Para definir todas las fases que abarcan la realización de los experimentos, vamos a adaptar la técnica conocida como GQM (*Goal Question Metric*) (Koziolek, 2008) a nuestros ensayos de laboratorio.

9.1. Goal Question Metric

Es una técnica que fue definida inicialmente en 1984 por *Basili* y *Weiss* (Basili and Weiss, 1984). Nos permite medir el proceso y los resultados de un proyecto. En principio fue desarrollada para evaluar procesos de desarrollo de productos o software, pero se puede extrapolar y aplicar fácilmente en los experimentos de laboratorio que haremos.

GQM se divide en tres niveles diferentes:

1. *Goals* (nivel conceptual): los objetivos nos marcan lo que queremos conseguir respecto a un producto, a un proceso o a un recurso. Ejemplos de objetivos que se pueden plantear: mejorar la calidad o la productividad, reducir costes, reducir riesgos ...

2. *Question* (nivel operacional): las preguntas se emplean para abordar cómo satisfacer un objetivo. La idea es caracterizar el objetivo con aspectos medibles.

3. *Metric* (nivel cuantitativo): buscar respuestas a las preguntas con un conjunto de datos.

El concepto que explica la idea principal es la siguiente: cualquier medición que se haga, tiene que hacerse orientada al objetivo (*Goals*) que se persigue. De esta manera el programa de medición será más preciso. Por tanto, el primer paso se corresponderá con la definición de un objetivo y, para comprobar que lo hemos alcanzado, tendremos que dar respuesta a una serie de preguntas (*Questions*). La información para contestar a dichas preguntas se extrae de las métricas (*Metrics*).

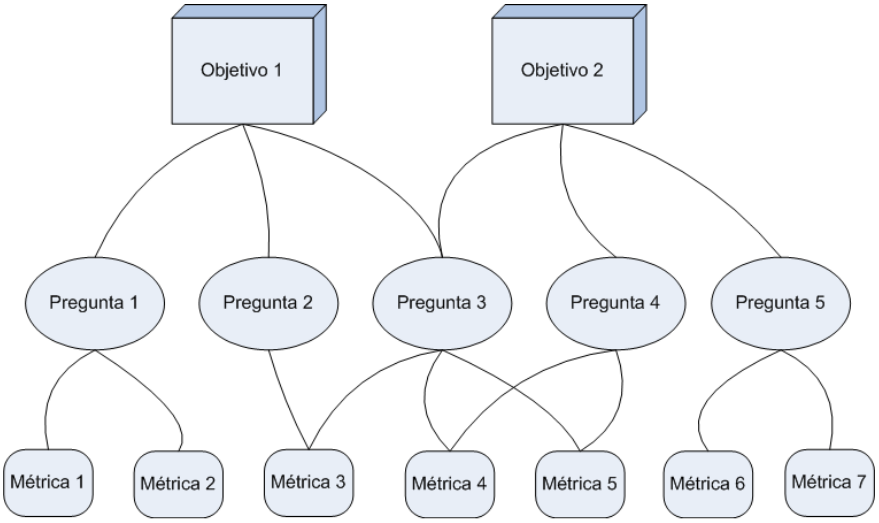


Figura 9.1: Esquema conceptual de GQM

En el esquema 9.1 se muestra la relación que puede existir entre las tres entidades involucradas: para cada objetivo pueden existir una o varias preguntas asociadas y a su vez, cada pregunta puede referenciar varios objetivos. Además, por cada pregunta puede haber más de una métrica. Y las métricas se pueden aplicar a más de una pregunta.

La aplicación de esta técnica se puede organizar en un proceso de 6 pasos: en los primeros se definen las métricas en base a los objetivos que se persiguen. En los últimos pasos se recogerán los datos y la información medida para calcular dichas métricas y comprobar si se ha cumplido el objetivo. A pesar de que GQM se define en un proceso de seis pasos, no se puede implementar como un proceso secuencial estricto. De hecho, muchos autores defienden que la implementación de GQM tiene que estar integrada con la planificación y gestión de un proyecto.

- Paso 1: definir los objetivos. En este paso se establecen varios puntos:
- ¿qué vamos a estudiar?
    - ¿cuál es la motivación y el propósito de las medidas?
    - ¿desde qué punto de vista se definen las metas?
    - ¿qué alcance tendrá? (entorno)

- Paso 2: generación de preguntas. Se formularán las preguntas que definan los objetivos marcados. Para comprobar si se ha alcanzado el objetivo es importante encontrar respuesta a dichas preguntas. En este paso vamos de un nivel conceptual a un nivel operacional por medio de las preguntas que nos planteemos. Por tanto, resulta evidente que las preguntas no pueden ser muy abstractas ya que entonces no encontraremos una relación entre las preguntas y las métricas.
- Paso 3: especificar las métricas. Las métricas se utilizan para responder a las preguntas planteadas en el paso anterior.
- Paso 4: preparar la recogida de datos. Se diseñará el sistema que permita recoger los datos necesarios que nos permitan evaluar las métricas. Es necesario determinar qué datos son necesarios para calcular las métricas del paso 3. También es preciso definir cómo se recogen esos datos. Lo ideal sería llevar a cabo un plan de medidas, en el que se especifique la descripción de las medidas, los resultados posibles, quién va a llevar a cabo cada medida, en qué momento y con qué herramientas.
- Paso 5: recogida de datos, validación y análisis de los mismos. Una vez recogidos los datos se comprueba su validez mediante el análisis de la información, verificando que sean correctos, estén completos y sean consistentes.
- Paso 6: analizar los datos: un análisis de los datos concluirá si se han cumplido los objetivos.

Podemos establecer una relación entre los distintos pasos que es necesario llevar a cabo en GQM y las fases de un proyecto.

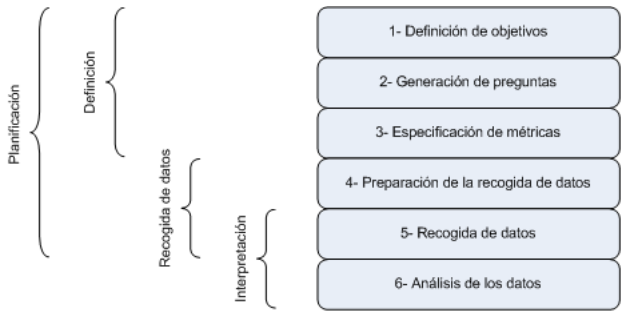


Figura 9.2: Fases de un proyecto y fases de GQM

En la Figura 9.2 vemos que los primeros pasos (1-5) se corresponden con una fase de planificación, donde se define y caracteriza el proyecto y se recoge toda la información necesaria para el inicio.

La fase de definición se correspondería con los pasos 1-3. En la definición, además de las preguntas y las métricas, se definen las medidas, indicando qué se va a medir, el propósito, con qué enfoque, el punto de vista y el contexto. Por último habrá que comprobar que las métricas definidas son consistentes y completas. Además, es necesario definir cuándo se deben realizar las mediciones y con qué herramientas.

La recopilación de datos corresponden con los pasos 4 y 5. En la recopilación de datos se recogen todas las métricas definidas.

La interpretación (donde se procesan los datos recopilados) corresponden con los pasos 5 y 6 del proceso de Basili. En la fase de interpretación se preparan los datos para su análisis y extracción de resultados, de manera que podamos obtener respuestas a las preguntas planteadas para saber si se ha cumplido el objetivo.

Para aplicar correctamente GQM habrá que fijar objetivos explícitos y especificarlos. Hay que evitar los casos de: " podemos obtener estos datos, vamos a ver qué podemos hacer con ellos". Además, cuando analicemos los datos, tenemos que tener en mente los objetivos, y no analizar los datos para ver qué comportamientos descubrimos.

También es importante tener en cuenta el contexto, considerando los factores de variación que puedan existir e identificando qué métricas responden a la pregunta formulada.

Además será necesario tener una infraestructura que soporte el sistema de mediciones, ya que la realización y almacenamiento de las mediciones supone un trabajo extra importante y es necesario tener controlados una gran cantidad de datos. Por último, debemos de ser conscientes de que las medidas no son el objetivo, si no una herramienta: no deberemos perder de vista el objetivo.

A continuación planteamos un ejemplo de adaptación del proceso GQM para nuestros ensayos de laboratorio y en los sucesivos capítulos aplicaremos esta técnica en experimentos concretos.

9.1.1. Aplicación de GQM (Goal-Questions-Metrics)

Como se ha comentado, este método se puede adaptar y aplicar en multitud de campos. En nuestro caso aplicaremos la técnica de GQM para desarrollar los ensayos de laboratorio, analizar los datos recogidos y extraer resultados y conclusiones. El objetivo que se pretende con la aplicación de GQM a los experimentos es facilitar la tarea de repetición de los mismos. Este hecho es importante, ya que la preparación de un experimento requiere gran cantidad de tiempo para definir todos los elementos a tener en cuenta y los pasos a seguir.

A continuación se explica cómo podemos aplicar GQM a la realización de nuestros experimentos:

1. Definir objetivos: en esta fase se definen, a grandes rasgos, las características del experimento y lo que se pretende.

   a) ¿qué vamos a estudiar?: será necesario especificar un objetivo genérico, por ejemplo, análisis de servicios de audio, vídeo, bajo demanda...

   b) ¿cuál es la motivación y el propósito? Es decir, especificar para qué se hace el experimento (para evaluar, para mejorar...) y qué se pretende con ello (medir calidad de servicio, analizar una herramienta, evaluar un algoritmo, el consumo de recursos...).

c) ¿desde qué punto de vista se hace el análisis? Desde un punto de vista de usuario por ejemplo, o desde un punto de vista investigador.

d) ¿en qué entorno o contexto se realizará el experimento? en un escenario cableado o inalámbrico, con un medio congestionado o con recursos libres ...

2. Generar preguntas y definir métricas: en nuestro caso podemos unir el paso 2 y 3 de la técnica propuesta por Basili en uno solo. Por un lado tenemos que transformar los objetivos en preguntas a las que, obviamente, tendremos que dar respuesta. Por ejemplo: ¿la calidad del vídeo recibido en el cliente es buena? Para poder llevar a cabo la evaluación y comprobar que se han alcanzado los objetivos, estas preguntas tienen que tener un equivalente en métricas. Por ejemplo, para evaluar la calidad de los contenidos que recibe el cliente podemos emplear como métrica la PSNR (*Peak Signal to Noise Ratio*) (Huynh-Thu and Ghanbari, 2008), asumiendo que la calidad es buena a partir de un cierto umbral. Otra métrica posible que podemos emplear sería el porcentaje de paquetes perdidos.

3. Fase de preparación del escenario y de la recogida de datos: en esta fase hay que determinar qué datos necesitamos recoger para evaluar las métricas propuestas en la fase anterior. Pero para ello es necesario caracterizar el entorno, definiendo todos los parámetros y variables que sean significativos para el experimento. En algunas ocasiones, esos parámetros y variables tomarán un único valor, pero en otras situaciones podrán adquirir un rango de valores. En este último supuesto hay que hacer una realización del experimento para cada valor. Por ejemplo, si queremos evaluar un servicio de *streaming* en función del ancho de banda disponible en el canal: los parámetros se corresponderían con el tipo de servicio y las características de la línea, y la variable de entrada sería el modelo de variación del ancho de banda en la línea. Hay que tener en cuenta que los escenarios de a/v que planteamos tienen muchos elementos a considerar, y un estudio completo resultaría inviable. Por tanto, será necesario establecer simplificaciones, teniendo en cuenta únicamente aquellas cuestiones que tengan relación directa con los objetivos y las métricas.

   Una vez identificado el entorno podemos preparar un escenario que permita evaluar los objetivos, recogiendo todos aquellos datos que sean necesarios para calcular las métricas. En el proceso de recogida es necesario definir cómo, dónde y cuándo se recogen los datos. Por ejemplo si queremos evaluar el *jitter* en el cliente habrá que definir con qué herramienta se capturan los datos necesarios para su cálculo, en qué lugar del escenario se procede a capturar dichos datos y en qué momento.

   Por último, una vez que el entorno esté caracterizado (teniendo en cuenta los parámetros que influyen en el objetivo propuesto), una vez que se hayan definido las variables que hay que tener en cuenta para la realización de los experimentos y una vez que se haya implementado el *testbed* (un escenario que permita evaluar los objetivos.), es necesario realizar un proceso de validación para comprobar que todo el entorno de pruebas funciona correctamente y que la recogida de datos se lleva a cabo sin errores. Además también habrá que examinar que los datos recogidos contienen la información suficiente para la posterior evaluación de los objetivos.

4. Realización del experimento: se lleva a cabo el ensayo.

5. Recogida y validación de datos: obtener toda la información necesaria sobre el experimento.

6. Extracción de conclusiones. es posible que en el desarrollo de esta fase obtengamos resultados inesperados o contradictorios. En ese caso tendremos que hacer un análisis más detallado que puede llevarnos a la necesidad de repetir el experimento o redefinir alguna fase o realizar nuevos experimentos.

El diagrama de la Figura 9.3 muestra las diferentes fases de la metodología propuesta, expresadas en un flujo de trabajo basado en modelos BPMN (*Business Process Model and Notation*).

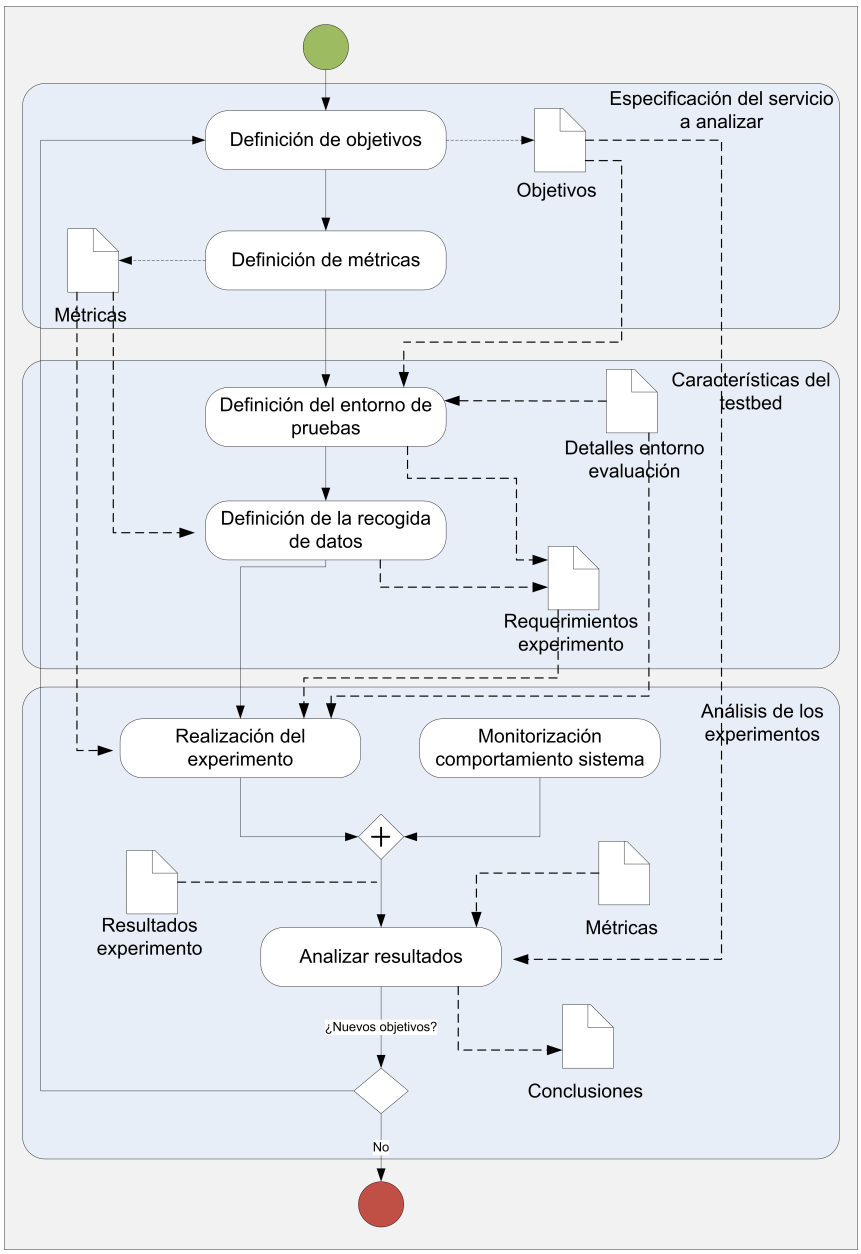


Figura 9.3: Metodología de los experimentos

9.2. Experimento 1: Evaluación de la calidad de los servicios streaming en diferentes condiciones de ocupación de red

A continuación planteamos un primer experimento en el que evaluaremos la calidad de los servicios *streaming* de vídeo cuando están expuestos a diferentes condiciones de ocupación de red.

Los servicios de envío de vídeo sobre redes IP son, al igual que la VoIP, altamente susceptibles a la degradación de la calidad. Dicha degradación puede ocurrir durante el proceso de codificación, durante la transmisión de los paquetes a través de la red y/o durante la decodificación y reproducción. En este experimento nos centraremos en la degradación de la calidad que ocurre a causa del proceso de transmisión de los contenidos a través de una red tipo *best-effort*.

En las redes de acceso, el ancho de banda disponible varía constantemente en función del número de usuarios activos. Además, los servicios de audio/vídeo tienen requerimientos de tiempo real, con lo cual son muy susceptibles a pérdidas o retardos en la red. Estos fenómenos pueden afectar negativamente a la calidad de las reproducciones en el cliente. En este primer experimento veremos cómo se degrada la calidad del servicio *streaming* en función de la congestión de la red y trataremos de reflejar la necesidad de disponer de un sistema adaptativo para la distribución de contenido multimedia.

9.2.1. Definición de objetivos

Estudiaremos servicios de vídeo *streaming* con contenidos de alta calidad, bajo una arquitectura simple de cliente-servidor en un entorno cableado, desde un punto de vista de investigación.

El objetivo será la evaluación de la calidad de las reproducciones en el cliente bajo distintas condiciones de ocupación en la red.

9.2.2. Generación de preguntas y definición de métricas

¿Cómo es la calidad de reproducción en un cliente que accede al servicio *streaming* con diferentes condiciones de disponibilidad en la línea?

Para evaluar la métrica de la calidad de vídeo podemos recurrir a métodos subjetivos (encuestas) u objetivos (empleando algoritmos para calcular el nivel de calidad).

La principal desventaja de los métodos subjetivos es el consumo de tiempo: es necesario mostrar el vídeo a un grupo de participantes y se les pide que puntúen la calidad del vídeo del 1 (inaceptable) al 5 (excelente). Estos valores de puntuación se conocen como escala MOS (*Mean Opinion Square*). Puesto que los resultados pueden variar de un test a otro a causa de su naturaleza subjetiva, es necesario contar con una población de más de 16 sujetos para realizar el experimento y obtener valores estables. Además también es importante controlar las condiciones en las que se lleva a cabo el test como, por ejemplo,

contar con un lugar tranquilo para realizar el experimento.

En las evaluaciones objetivas se emplean algoritmos, los cuales se pretende que se correlen lo máximo posible a las medidas de calidad de los métodos subjetivos. Hay tres tipos de algoritmos: *full-reference*, *no-reference* y *partial-reference*. El primero de ellos compara el vídeo en el origen y en el destino y determina el nivel de distorsión; para el segundo algoritmo sólo se analiza el *stream* a la salida; y para el último se extraen parámetros específicos del *stream* original y se comparan con los mismos parámetros del *stream* recibido (Telchemy, 2008).

El algoritmo más utilizado como métrica para la evaluación de la calidad de un vídeo es la PSNR (*Peak Signal to Noise Ratio*), un algoritmo tipo *full-reference* en el que se realiza una comparación detallada entre el *stream* original y el *stream* que se recibe en el cliente. Hay multitud de publicaciones que consideran la PSNR como una métrica relevante en cuanto a evaluación de la calidad. Por ejemplo, (Ennaji et. al, 2009) cuantifican la degradación de la calidad cuando la sesión *streaming* tiene lugar en redes WLAN (*Wireless Local Area Network*) a causa de la distancia, de los obstáculos o del movimiento de los receptores. La PSNR es un algoritmo costoso computacionalmente hablando. Su expresión teórica es la siguiente:

$$PSNR = 20 \cdot log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right) \tag{9.1}$$

siendo MAX_I el máximo valor de un pixel en la imagen (si cada pixel se representa con un número determinado de bits, B, entonces $MAX_I = 2^B - 1$) y definiendo el MSE (*Mean Square Error*, error cuadrático medio) como

$$MSE = \frac{1}{M \cdot N} \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} \parallel I(i,j) - K(i,j) \parallel^2 \tag{9.2}$$

donde I, K representan dos imágenes de tamaño MxN.

A partir de los valores de la PSNR se puede establecer una relación con los valores de la escala MOS (ver Tabla: 9.1). La escala MOS nos proporciona valores numéricos que indican la calidad recibida de una manera mucho más clara (Klaue et. al, 2003). Un valor de PSNR de 35 dB se considera bueno, mientras que por debajo de 20 dB es inaceptable.

PSNR [dB]	MOS
> 37	5 (Excelente)
31 - 37	4 (Buena)
25 - 31	3 (Aceptable)
20 - 25	2 (Pobre)
< 20	1 (Mala)

Tabla 9.1: Escala MOS para valores de PSNR

Por tanto, parece bastante acertado emplear la escala MOS como métrica para este primer experimento: conociendo los valores MOS de las reproducciones podremos determinar

SSIM	MOS
$> 0{,}99$	5 (Excelente)
$\geq 0{,}95$ & $< 0{,}99$	4 (Buena)
$\geq 0{,}88$ & $< 0{,}95$	3 (Aceptable)
$\geq 0{,}5$ & $< 0{,}88$	2 (Pobre)
$< 0{,}5$	1 (Mala)

Tabla 9.2: Escala MOS para valores de SSIM

si la calidad del servicio *streaming* resulta aceptable, completando así el objetivo marcado.

Una segunda métrica, también del tipo *full-reference,* es el SSIM (*Structural Similarity Index Metric*), que mide la similitud entre dos imágenes. El valor máximo que puede tomar la métrica SSIM es '1.0', indicando que las imágenes comparadas son idénticas. (Zinner et. al, 2010) plantean una clasificación de los valores SSIM en la escala MOS (Tabla 9.2). Sin embargo, la PSNR sigue siendo la métrica más ampliamente utilizada para medir la QoE.

Ambas métricas, PSNR y SSIM, están muy relacionadas, pudiendo llegar a predecir los valores de PSNR a partir de los valores de la métrica SSIM, y viceversa (Hore and Ziou, 2010). Sin embargo, a pesar del uso ampliamente extendido de la PSNR como métrica de evaluación de la calidad, ésta presenta una serie de limitaciones que deberemos considerar a la hora de emplear la métrica de la PSNR en los sucesivos escenarios y experimentos que llevaremos a cabo a lo largo de los próximos capítulos (Huynh-Thu and Ghanbari, 2012). La cuestión más importante a tener en cuenta es el hecho de que la PSNR se calcula *frame* a *frame*, estableciendo una relación entre el *frame* en el origen y el *frame* en el destino. En nuestro caso, en experimentos futuros, trabajaremos con escenarios que presentan pérdidas de *frames* o reducciones en el *frame rate* con respecto al vídeo original. Es por eso que, antes de procedecer al cálculo de la PSNR, deberemos asegurarnos de que el vídeo original y el vídeo recibido presentan el mismo número de *frames*, supliendo la pérdida de *frames* en recepción con la copia del *frame* anterior recibido. Este planteamiento es el mismo que implementan ciertos reproductores, donde la ausencia de *frames* durante el visionado se traduce en un congelado de la imagen.

9.2.3. Fase de preparación del escenario y de la recogida de datos

Para caracterizar el entorno y poder proceder a la preparación del escenario deberemos definir e identificar todos aquellos aspectos que sean significativos a la hora de estudiar un servicio de *streaming* bajo diferentes condiciones de ancho de banda disponible en la red de acceso.

Para el estudio que se plantea será necesario reproducir una arquitectura cliente-servidor de servicios de audio/vídeo en Internet en un entorno controlado de pruebas. Para ello, podemos reproducir unas condiciones de red cambiantes en una maqueta y emplear técnicas de emulación o simulación. Lo ideal, siempre que sea posible, es recurrir a pruebas basadas en *testbed* donde se emulan los comportamientos de algunos o todos los componentes de la maqueta. Si no fuese posible, (debido a elevados tiempos de duración

del experimento, o un número de usuarios muy elevado) nos decantaremos por la simulación.

Para evaluar diferentes condiciones de ocupación de red, una posible solución es emplear algoritmos de conformado de tráfico, como por ejemplo *Token Bucket*, gracias al cual es posible controlar la cantidad de datos que se inyectan en una red. El conformado de tráfico y políticas de calidad de servicio se pueden aplicar directamente sobre las tarjetas de red de un equipo bajo un sistema operativo *Linux* (con el módulo TC - *traffic control*[1]). Otra posible opción para emular distintas capacidades en la línea de acceso es utilizar el emulador NS-3[2], que además nos permite inyectar tráfico real, proveniente de nuestras aplicaciones, en el entorno de emulación, de manera que podemos estudiar distintas condiciones de congestión sin necesidad de desplegar físicamente las redes.

En la primera fase de definición de objetivos hemos optado por una arquitectura simple de cliente-servidor. En nuestro caso será importante considerar las plataformas de *streaming* que utilizaremos, puesto que el experimento se realizará con usuarios reales que van a acceder a un servidor en concreto. Como servidor *streaming* emplearemos las librerías de *live555*[3]. En el caso del cliente utilizaremos *openRTSP* para acceder al servicio de *streaming*. Ambas herramientas presentan una solución de software libre y, para este experimento, no realizaremos modificaciones sobre los mismos.

Un aspecto que no es significativo para el experimento es la red troncal, ya que asumimos que las limitaciones de ancho de banda se encuentran en la red de acceso de los clientes. Tampoco tendremos en cuenta el tráfico adicional de otros servicios para este estudio.

Por otro lado, un factor clave será la capacidad de ancho de banda disponible en la línea. Podemos expresar esta disponibilidad como el cociente entre el ancho de banda de la línea y la tasa a la que se sirven los contenidos al cliente, definiendo así una variable 'q' que será clave a lo largo de los siguientes experimentos:

$$q = \frac{BW}{Tasa\,de\,codificación} \tag{9.3}$$

A efectos prácticos, es inmediato comprobar que un valor de q<1 indicaría una situación de congestión, ya que la tasa a la que se sirven los contenidos es mayor que el ancho de banda disponible. Análogamente, un valor de q>1 se traduce en situaciones en las que existen recursos libres en la red y no hay congestión.

Ese parámetro 'q' será nuestra variable de entrada y tomará valores dentro del intervalo [0.7 - 1.1], de manera que abarcamos escenarios en los que la línea esté ocupada al 100 % y en los que hay recursos libres.

Para poder evaluar la métrica propuesta de la PSNR tendremos que almacenar el vídeo original y el vídeo recibido por el cliente. Con estos dos datos, y haciendo uso de una

[1]Linux Advanced Routing and Traffic Control, http://lartc.org
[2]http://www.nsnam.org/
[3]http://www.live555.com/

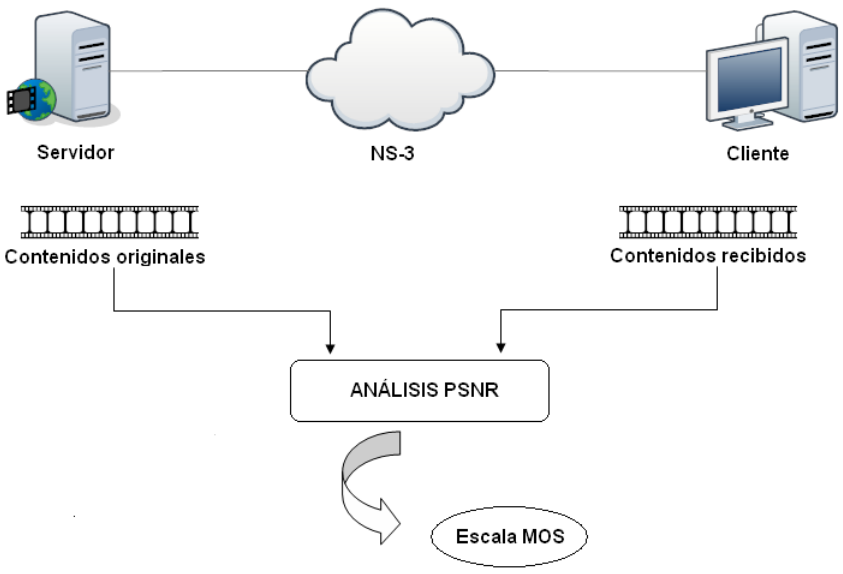


Figura 9.4: Experimento 1

herramienta de análisis de la PSNR, podemos identificar el valor obtenido con la métrica de la PSNR en la escala MOS.

Antes de proceder a la realización de los experimentos es necesario comprobar y validar el *testbed*: hay que comprobar que el porcentaje de uso de CPU (*Central Processing Unit*) y memoria de los equipos no supera el 80 %, con el fin de asegurarnos de que los equipos están funcionando dentro de los límites aceptados.

También se comprueba, mediante *iperf* [4], que el ancho de banda disponible en cada instante concuerda con el especificado en el modelo de red de NS-3. *Iperf* es una herramienta que nos permite medir el ancho de banda de un enlace entre dos equipos. Su modo de funcionamiento entraría dentro de los sistemas intrusivos que explicamos en la sección 8.1, ya que el cálculo del ancho de banda se hace enviando paquetes TCP o UDP y analizando latencias, *jitter* o pérdidas de ese tráfico introducido. En nuestro caso resultará útil para comprobar el ancho de banda que la herramienta de emulación de red NSE está ofreciendo. La Figura 9.4 resume las principales características de la fase de preparación del escenario y la recogida de los datos.

9.2.4. Realización del experimento

Una vez implementado el escenario y el sistema de recogida de datos (en este caso bastará con almacenar el vídeo recibido en el cliente y el vídeo original del servidor), se procede a realizar el experimento.

Para cada valor del cociente de disponibilidad 'q' haremos 10 realizaciones, con el objetivo de obtener valores estables y que los resultados no se vean afectados por situaciones anómalas.

[4] https://iperf.fr/

9.2.5. Recogida y validación de los datos

Para poder evaluar la métrica de la PSNR necesitamos el vídeo en el origen y destino. Para cada *frame* recibido obtendremos un valor de PSNR. Ese valor de PSNR tendrá una correspondencia con un valor de la escala MOS según la Tabla 9.1. Para más información sobre la realización detallada del experimento y la recogida y validación de los datos ver el apéndice D.

9.2.6. Extracción de conclusiones

En la gráfica 9.5 vemos los resultados en la escala MOS para los diferentes escenarios que planteamos en cuanto a niveles de ocupación en la red.

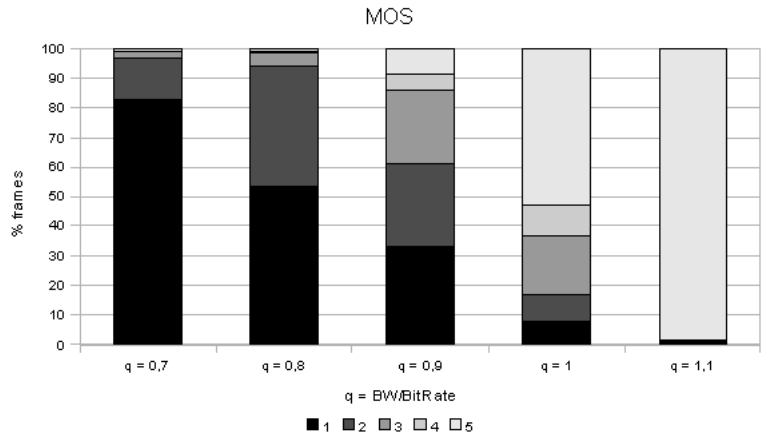


Figura 9.5: Resultados valores MOS en diferentes situaciones de congestión

Se observa que, a medida que aumenta la congestión (valores de 'q' menores que 1), aumenta el porcentaje de *frames* que se corresponden con los niveles más bajos de calidad en la escala MOS, indicativo de una calidad muy pobre en el vídeo recibido. Por ejemplo, cuando el ancho de banda consumido por la sesión *streaming* supera en un 30 % el ancho de banda de la línea (q=0.7), más del 80 % de los *frames* que se reciben en el cliente tienen una calidad clasificada como "mala" o "inaceptable" en la escala MOS. Por el contrario, cuando la capacidad de la red puede hacer frente al ancho de banda consumido por el servicio multimedia (q>1), prácticamente la totalidad de los *frames* recibidos reciben una calificación de excelente, lo que se traduce en una buena experiencia de usuario.

La explicación de este fenómeno es bastante sencilla e intuitiva: recordemos que un vídeo es una secuencia de imágenes o *frames*. Hay tres tipos básicos de *frames* codificados: *intra-coded frames* (I-*frames*): los *frames* se codifican independientemente de otros *frames*); *predictively codec* (P-*frames*): el *frame* es codificado basándose en un *frame* previo; y *bi-directionally predicted frames* (B-*frames*): el *frame* se codifica usando tanto *frames* previos como futuros (ver sección5.1.1). Antes de proceder al envío, es necesario llevar a cabo un proceso de segmentación y empaquetado. Los *frames* de referencia (tipo I) serán los que necesiten mayor número de paquetes para su transmisión, por lo tanto, la probabilidad de perder paquetes correspondientes a un *frame* tipo I es mucho mayor que para

(a) Escenario sin pérdidas - *frame* 48

(b) Escenario con pérdidas - *frame* 48

(c) Escenario sin pérdidas - *frame* 150

(d) Escenario con pérdidas - *frame* 150

Figura 9.6: Ejemplos de pérdida de calidad

el resto de *frames*.

La pérdida de un único paquete perteneciente a un *frame* conlleva a errores durante el proceso de decodificación, pudiendo afectar severamente a la calidad del vídeo. Sin embargo, no todas las ocurrencias de pérdidas de paquetes tienen el mismo impacto sobre la calidad percibida. Por ejemplo, si ocurre un error en un *frame* tipo I o P se propagará a los restantes B o P y es probable que las pérdidas se hagan visibles en el vídeo durante algunos segundos en algunos casos. Sin embargo, un error en un *frame* B no se propagará a los subsecuentes *frames* y puede no ser detectable por el receptor (ver sección 5.2).

Para completar los resultados y las explicaciones de este estudio vamos a mostrar ejemplos de *frames* capturados en el receptor. La secuencia de vídeo empleada para estas capturas es la conocida secuencia de *Foreman*, empleada en diversos estudios sobre la calidad de las imágenes[5].

Las Figuras 9.6b y 9.6d presentan un ejemplo de pérdida de calidad en el vídeo. En el primer caso, la degradación de la calidad no es tan acusada como en el segundo, puesto que la imagen se corresponde con una secuencia estática, donde los cambios entre imágenes consecutivas se centran en la región de la boca del personaje. Dichas variaciones pueden ser codificadas con frames tipo P o B. El segundo caso, donde la secuencia presenta mayor grado de movimiento que la primera, es fruto de la pérdida sucesiva de *frames* tipo I y tipo P.

[5]http://trace.eas.asu.edu/

Capítulo 10

ESTIMACIÓN Y ADAPTACIÓN EMPLEANDO TÉCNICAS DE TRANSCODIFICACIÓN

Desde la aparición de la tecnología *streaming* (a mediados de los años 90), diversos tipos de compañías, instituciones y medios de comunicación han desplegado servicios de vídeo en Internet. A lo largo de todo este tiempo, la experiencia de usuario ha ido mejorando a la par que mejoraba la tecnología que da soporte a estos servicios. Desgraciadamente, aún persiste un problema: cómo adaptar la calidad del vídeo y, en definitiva, el consumo del ancho de banda, a un entorno heterogéneo como Internet.

A pesar de la existencia de ciertas mejoras (ver capítulo 8), aún existen puntos débiles en la arquitectura de estos servicios, especialmente en lo que se refiere a la estimación del ancho de banda. Este hecho resulta crucial, puesto que para definir correctamente el *bitrate* de transmisión de un vídeo es necesario conocer cuántos recursos hay disponibles. Esto sólo puede ser posible analizando qué está ocurriendo en la red durante el proceso de transmisión del vídeo.

Conscientes de este problema, en este capítulo proponemos un estimador no instrusivo para servicios de vídeo en Internet. Para ello, hemos hecho modificaciones en aplicaciones *streaming* de código libre (*Darwin Streaming Server* y *VLC player*) con el fin de proporcionar las capacidades de estimación de ancho de banda y adaptación de los contenidos. Nuestra versión modificada del *Darwin Streaming Server* procesa los paquetes de control provenientes del cliente con la información necesaria para determinar el valor del bitrate óptimo y lo envía a la herramienta de codificación en tiempo real *VLC Player*. Con el simple análisis de la llegada de los paquetes en el cliente, el estimador es capaz de decidir si es necesario incrementar el bitrate del vídeo, reducirlo o mantenerlo, de acuerdo al ancho de banda disponible extremo-a-extremo. El sistema se puede introducir fácilmente en una arquitectura típica de un servicio de vídeo *streaming* produciendo además una sobrecarga en el tráfico mínima (los paquetes de *feedback* representan menos del 0.014 % del número total de bytes transmitidos).

A lo largo de este capítulo se describirá el proceso de diseño, implementación, evaluación y mejora del sistema de vídeo adaptativo propuesto basado en técnicas de trans-

codificación. Para ello, se llevaron a cabo una serie de experimentos de laboratorio y la evaluación del estimador se realiza mediante diferentes test con diferentes cargas en la red. Los resultados muestran que el sistema adaptativo implementado es capaz de llevar a cabo labores de estimación y adaptación del bitrate de manera precisa. Sin embargo, el consumo de recursos en el servidor es el principal factor limitante de esta propuesta.

10.1. Experimento 1: Evaluación de parámetros característicos del tráfico *streaming* en diferentes condiciones de ocupación de red

En la primera tanda de experimentos trataremos de caracterizar los parámetros más significativos para una sesión de *streaming*. Como hemos visto en la Parte II de la documentación, para la transmisión de contenidos multimedia a través de una red de conmutación de paquetes (como lo es Internet) es necesario fragmentar los contenidos para posteriormente enviarlos al cliente en paquetes RTP. La transmisión de dichos paquetes por líneas que no implementan calidad de servicio hace que las condiciones de la red tengan una influencia significativa sobre las características del tráfico recibido en el cliente. Por ejemplo, los paquetes se verán afectados de diferente manera por los procesos de encolado de los elementos intermedios de la red. Este hecho hace que los contenidos recibidos en el cliente difieran de los contenidos originales a causa de las pérdidas o los retardos que puedan producirse durante la transmisión.

La idea que está detrás de este experimento es encontrar unos patrones o tendencias en el comportamiento de diversos parámetros ante diferentes situaciones del nivel de congestión. El análisis exhaustivo del tráfico *streaming* marcará las directrices para la posterior mejora de un método de estimación no intrusivo, en el que, únicamente analizando el tráfico que llega al cliente, se pueda extraer información sobre la congestión de la red, pudiendo así realizar tareas de adaptación de los contenidos.

10.1.1. Definición de objetivos

Estudiaremos servicios de vídeo *streaming* con contenidos de alta calidad, bajo una arquitectura simple de cliente-servidor en un entorno cableado, desde un punto de vista de investigación con el objetivo de evaluar el comportamiento de parámetros característicos de una sesión *streaming* cuando las condiciones de ocupación de la red varían.

10.1.2. Generación de preguntas y definición de métricas

¿Qué parámetros característicos de un flujo de vídeo *streaming* varían en función de la congestión de la red?

Los servicios de *streaming* de audio/vídeo tienen requisitos de tiempo real y como tal, se ven afectados por la pérdida de paquetes y el *jitter* en la red (ver capítulo 7). Pero además de estas métricas clásicas, incluimos la evaluación de la separación media entre paquetes y, como novedad, proponemos el análisis de una nueva métrica basada en la linealidad de los instantes de recepción. A continuación se describe cada una de ellas.

- La pérdida de paquetes es una métrica ampliamente utilizada para detectar situaciones de congestión. Este fenómeno tiene lugar cuando la tasa de transmisión supera al ancho de banda disponible en la línea (haciendo referencia a la fórmula 9.3, se correspondería con cualquier situación en la que q<1).

- *Jitter*: para poder transmitir los contenidos multimedia a través de una red de datos es necesario llevar a cabo un proceso de segmentación, donde los *frames* que forman el vídeo se dividen en paquetes. En las redes de conmutación de datos cada paquete puede recorrer caminos distintos para alcanzar el destino y por lo tanto, cada paquete puede sufrir retardos distintos. Esa variación de retardo se conoce como *jitter*. Como hemos explicado en el apartado 7.3, el *jitter* afecta especialmente al tráfico con requerimientos de tiempo real.

  El *jitter* en el instante i puede calcularse a partir de la expresión:

$$J_i = J_{i-1} + \frac{\mid D(i-1,i) - J_{i-1} \mid}{16} \tag{10.1}$$

  Siendo

$$D(i,j) = (R_j - S_j) - (R_i - S_i) \tag{10.2}$$

  donde R representa el instante de llegada y S es el *timestamp* RTP.

  Para explicar la fórmula anterior del *jitter* podemos recurrir al siguiente esquema:

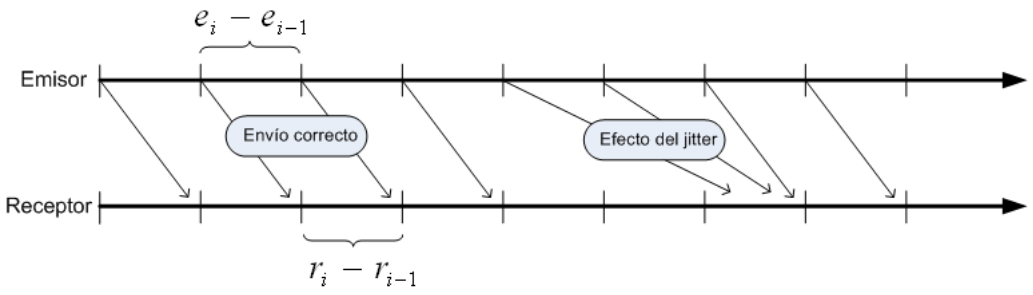


Figura 10.1: Esquema para el cálculo del jitter

  En la figura anterior (10.1), podemos expresar el *jitter* instantáneo como la diferencia que existe en el tiempo entre paquetes consecutivos en el receptor comparado con el tiempo entre paquetes en el emisor.

$$jitter_{instantaneo} = |(r_i - r_{i-1}) - (e_i - e_{i-1})| \tag{10.3}$$

  A partir de la expresión anterior es inmediato deducir que, en una situación ideal, el *jitter* será nulo. Esta expresión de *jitter* instantáneo (10.3) es equivalente a la diferencia en el "tiempo de tránsito relativo" para dos paquetes, que se puede calcular

como la diferencia entre el *timestamp* del paquete RTP y el instante de recepción. De ahí la expresión 10.2.

De esta manera, el *jitter* , que se define como una estimación de la varianza estadística del tiempo entre llegadas de paquetes de datos RTP, quedaría definido como:

$$jitter_{actual} = jitter_{previo} + \frac{jitter_{instantaneo} - jitter_{previo}}{16} \tag{10.4}$$

siendo esta fórmula idéntica a la expresión 10.1 para el cálculo de esta métrica.

- La linealidad (coeficiente de correlación de *Pearson*, (Rodgers and Nicewander, 1988)): a consecuencia del proceso de segmentación y empaquetado, cada *frame* es dividido en varios paquetes. Todos los paquetes que pertenecen a un mismo *frame* se transmiten en el mismo instante y con la misma marca temporal (*timestamp*). Por tanto, el patrón del tráfico del servidor responde a un modelo a ráfagas.

  En la figura 10.2 vemos un ejemplo de distribución del tráfico a la salida del servidor donde apreciamos claramente ese modelo a ráfagas. La separación entre ráfagas vendrá marcada por el factor de los *frames* por segundo (fps). Así mismo, el número de paquetes que forman cada ráfaga dependerá del tipo de *frame*. De esta manera, tal y como explicamos en el capítulo 5.2, los frames tipo B o P sufren mayor compresión, por lo que serán necesarios menos paquetes para su transmisión.

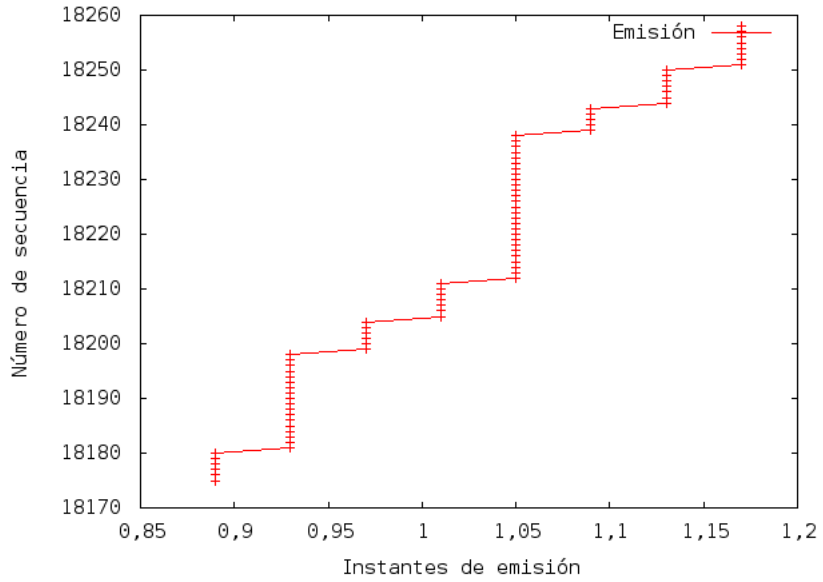
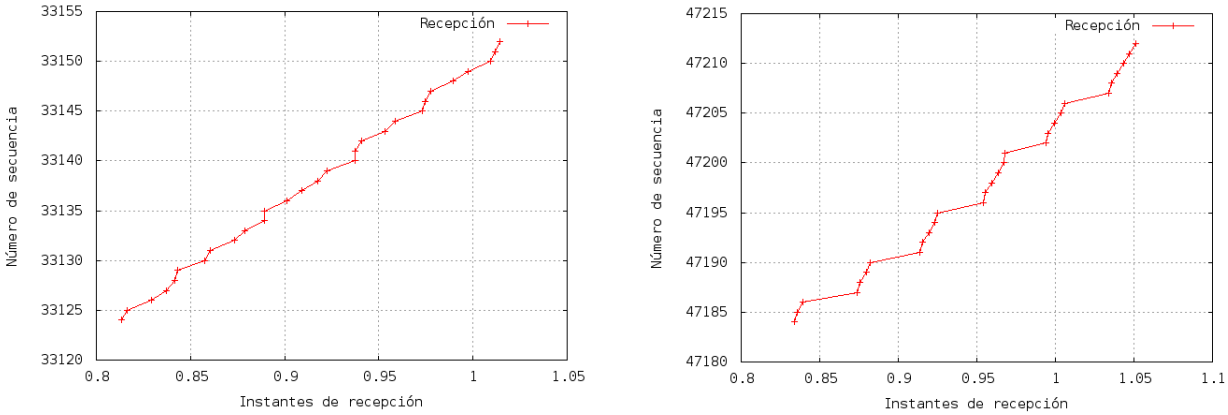


Figura 10.2: Patrón de emisión

Sin embargo, durante el proceso de transmisión pueden ocurrir retardos derivados de situaciones de congestión en la línea. Este hecho provoca que los paquetes no lleguen a ráfagas y que la curva de recepción presente un aspecto lineal, no pudiendo distinguir tan claramente las ráfagas a causa de esta dispersión temporal (figura 10.3a).



(a) Con congestión (b) Sin congestión

Figura 10.3: Patrón de recepción

Por el contrario, cuando no hay congestión en la red, el patrón de tráfico recibido se asemejará mucho al patrón a ráfagas del servidor (figura 10.3b).

Por lo tanto, podemos analizar la linealidad en los paquetes de llegada para detectar situaciones de congestión en la red. Para ello emplearemos el coeficiente de correlación de *Pearson*, R. Este índice estadístico permite medir la relación lineal entre dos variables (X e Y) y además es independiente de la escala de medida. Cuando R = 0 quiere decir que no existe una relación lineal entre las dos variables. Sin embargo, cuando R = 1 implica que al aumentar una variable la otra aumenta exactamente en la misma proporción, con lo que habrá una correlación perfecta. Su expresión matemática es:

$$R_j = \frac{\sum X_j \cdot Y_j - \frac{1}{N} \cdot \sum X_j \cdot Y_j}{\sqrt{(\sum X_j^2 - \frac{1}{N} \cdot (\sum X_j)^2) \cdot (\sum Y_j^2 - \frac{1}{N} \cdot (\sum Y_j)^2)}} \tag{10.5}$$

donde X_j será el vector de la variable X durante el instante j, Y_j el vector de la variable durante el instante j y N el tamaño de los vectores.

- GAP: se define como la separación media entre paquetes. A la salida del servidor, la separación media entre paquetes pertenecientes al mismo *frame* es prácticamente nula. En un escenario ideal, sin congestión, el tiempo de recepción entre paquetes del mismo *frame* será muy pequeño y entre paquetes de distinto *frame* se correspondería con el valor de 1/fps. Sin embargo, cuando aumenta la congestión, la separación entre paquetes del mismo *frame* va en aumento a causa de los efectos de la dispersión temporal. Este efecto será también el causante de que el tiempo entre paquetes consecutivos de distinto *frame* se vea disminuido.

  El valor de la separación media entre paquetes, T, en un segundo 'j' vendrá dado por la siguiente expresión:

$$T_j = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_{i-1}) \tag{10.6}$$

donde x_i y x_{i-1}representan los instantes de recepción de dos paquetes consecutivos que pertenecen al mismo *frame*, y N representa el número de paquetes recibidos en un segundo.

10.1.3. Fase de preparación del escenario y de la recogida de datos

El escenario general será el mismo que para el experimento 1 explicado en la sección 9.2.3. Recurriremos a un *testbed*, formado por un equipo servidor, un equipo cliente y un tercer equipo emulando la línea de acceso. Este último se configurará para emular diferentes capacidades de canal, empleando el software NS-2 [1]. Concretamente, se ha establecido que el ancho de banda del canal tomará valores del intervalo [700kbps - 5.25 Mbps], mientras que la tasa de transmisión de los contenidos oscilarán entre [1 Mbps - 3,5 Mbps]. Con este rango de valores en las variables de entrada podemos obtener un rango de variación del parámetro 'q' (ecuación 9.3) entre [0.7 y 1.5]. Este intervalo de valores nos da la posibilidad de comprobar la fiabilidad de las métricas en diferentes condiciones de congestión. Recordemos que, una ocupación de la línea del 100 % viene indicada por valores q<1 y, por ejemplo, q=1.5 indicarían una ocupación de línea del 0.66 %.

Entre el equipo cliente y el servidor se efectuará una sesión *streaming*. Será necesario disponer de las trazas de los paquetes en el servidor y en el cliente. Para ello emplearemos la herramienta *tshark*[2], que nos permite capturar el tráfico de la sesión y analizar las cabeceras de los paquetes, pudiendo así obtener los instantes de emisión y recepción del paquete, IP origen y destino y, para el caso concreto del protocolo RTP, los números de secuencia y los *timestamps*. Una vez capturado el tráfico de la sesión dispondremos de los datos necesarios para calcular las métricas.

- Para el cálculo del porcentaje de pérdidas necesitamos disponer del número total de paquetes enviados y los recibidos.
- Para calcular el *jitter* emplearemos la fórmula del *jitter* descrita en el RFC de RTP (RFC3550) y expuesta en la ecuación 10.1.
- Para calcular el gap necesitamos los instantes de recepción en el cliente de paquetes consecutivos pertenecientes a un mismo *frame*, esto es, paquetes consecutivos que presenten el mismo valor de *timestamp*.
- Para el cálculo de la linealidad recurrimos a la fórmula del coeficiente de correlación de *Pearson*, R, que responde a la ecuación 10.5, donde las variables serán el tiempo de llegada de los paquetes y el número de secuencia. Por tanto, para este caso: X_j será el vector de tiempos de recepción durante el segundo j, Y_j el vector de los números de secuencia durante el segundo j y N el tamaño de los vectores. Como ya se ha comentado, un valor de R próximo a la unidad indica una relación lineal entre ambas variables.

[1] http://www.isi.edu/nsnam/ns/
[2] http://www.wireshark.org/docs/man-pages/tshark.html

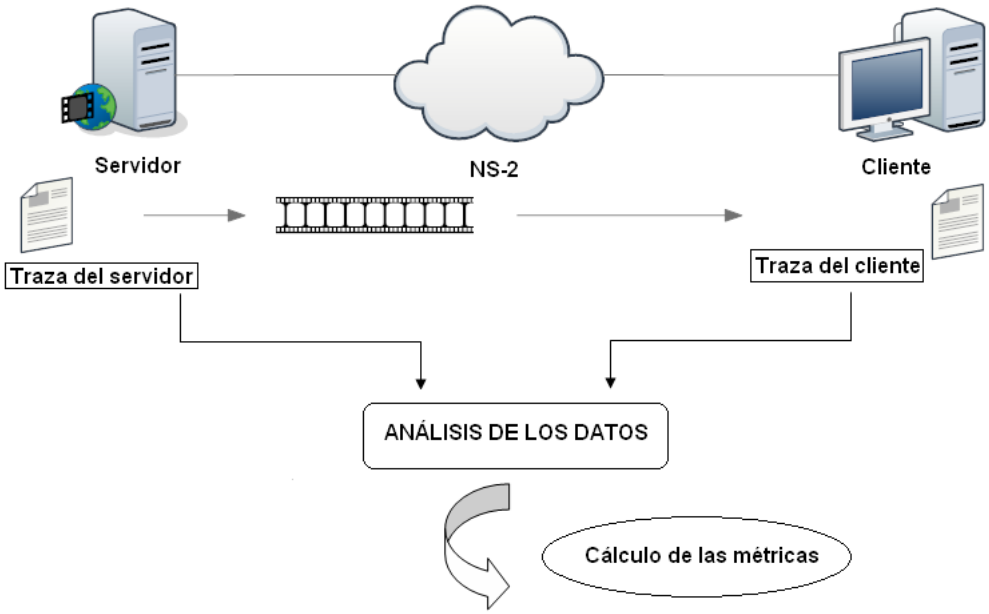


Figura 10.4: Experimento 1

Al igual que en el apartado 9.2.3, se comprueba y se valida todo el sistema antes de proceder con los experimentos.

10.1.4. Realización del experimento

Se procede a la realización de las pruebas, con varias repeticiones para cada combinación de la tasa de codificación de los contenidos y el factor de disponibilidad 'q'.

10.1.5. Recogida y validación de los datos

Una vez finalizada la realización del experimento procederemos al análisis de los datos recogidos con la herramienta *tshark*.

10.1.6. Extracción de conclusiones

En las gráficas sucesivas podremos ver el resultado de las métricas para los distintos niveles de congestión. Como iremos viendo en los análisis, las métricas estudiadas están muy relacionadas con el nivel de congestión en la red. Por tanto, podrán ser utilizadas como parámetros de control en el algoritmo de adaptación que propondremos en sucesivos experimentos.

10.1.6.1. Pérdidas

A continuación analizamos el porcentaje de pérdidas para diferentes valores de disponibilidad en el enlace y para distintas tasas de codificación de los contenidos de vídeo que se transmiten. Como vemos en la figura 10.5, cuando estamos en una situación sin congestión (q>1) las pérdidas de paquetes son inferiores al 1 % para todas las calidades

de vídeo analizadas. Según los estudios realizados por (Wu et. al, 2000) este porcentaje de pérdidas no afecta en gran medida a la calidad del vídeo recibido, ya que el análisis de la PSNR devuelve valores en torno a 27 dB en el peor de los casos, y concluyen que es un indicador de buena calidad.

Cuando el nivel de ocupación en la red excede en un 10 % la capacidad del enlace, se producen unas pérdidas cercanas al 10 % de los paquetes. Cuando el exceso de tráfico es de un 20 % las pérdidas se sitúan en torno a un 20 % y así sucesivamente.

De acuerdo estos datos, esta métrica se puede utilizar para detectar situaciones de congestión graves, en las que tienen lugar pérdida de paquetes.

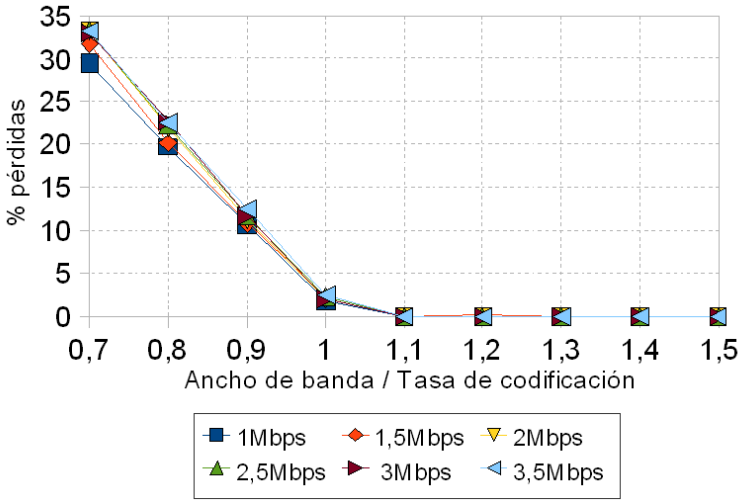


Figura 10.5: Porcentaje de pérdidas para distintos niveles de ocupación y distintas calidades de los contenidos

10.1.6.2. Jitter

La siguiente métrica que analizaremos será el *jitter*.

En la figura 10.6 apreciamos una clara disminución en los valores de *jitter* a medida que la ocupación en la red disminuye. El *jitter* es un indicador claro de congestión en la red: a medida que se incrementa la saturación en las líneas de acceso, se incrementan también los tiempos de encolado en los elementos intermedios de la red. Es, por tanto, una métrica muy interesante para detectar congestiones de manera temprana, antes incluso de que tenga lugar la pérdida de paquetes.

10.1.6.3. Linealidad

A continuación evaluamos la variación entre los tiempos de salida y los tiempos de llegada como indicador de congestión. En un caso ideal, el patrón de llegada de paquetes al cliente será prácticamente igual al patrón de salida del servidor. Por el contrario, cuando una red presente congestión, los patrones de salida y llegada serán diferentes a causa de los procesos de encolado en los elementos intermedios. Por lo tanto, esta medida puede

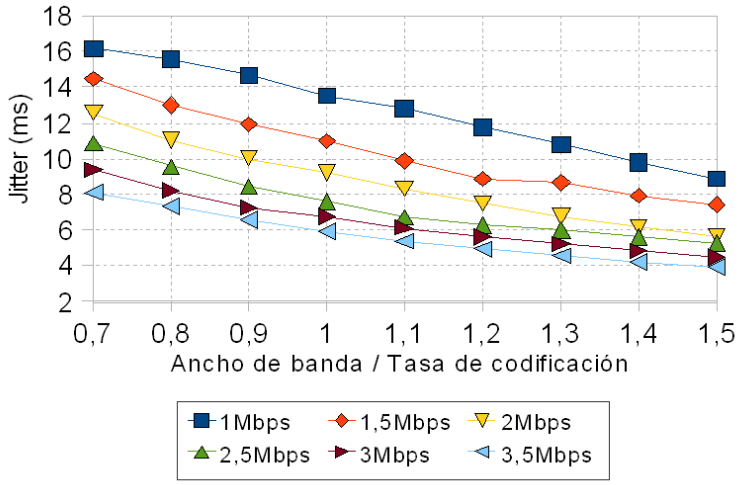


Figura 10.6: Jitter para distintos niveles de ocupación y distintas calidades de los contenidos.

ser una métrica de estimación de congestión.

Los resultados del análisis se muestran en la figura 10.7.

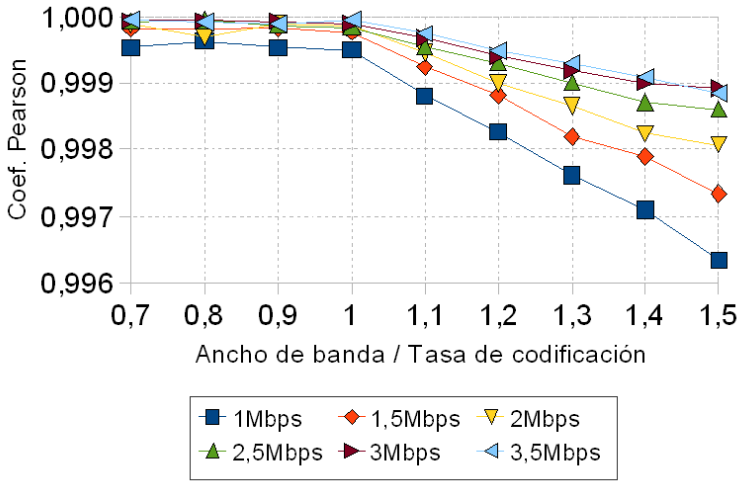


Figura 10.7: Linealidad en los tiempos de recepción para distintos niveles de ocupación y distintas calidades de los contenidos

Las curvas para cada uno de los valores de la tasa de codificación presentan un comportamiento similar. Podemos observar que el valor del coeficiente de correlación de *Pearson* presenta valores muy próximos a la unidad en situaciones de congestión (q<1). A medida que disminuye la congestión en la red también disminuye la linealidad en la recepción de los paquetes, por lo que el tráfico recibido presenta un patrón a ráfagas y, consecuentemente, el coeficiente de linealidad disminuye.

10.1.6.4. Gap

En cuanto a la separación entre paquetes también hemos observado un patrón de comportamiento muy similar para los distintos vídeos empleados en el experimento.

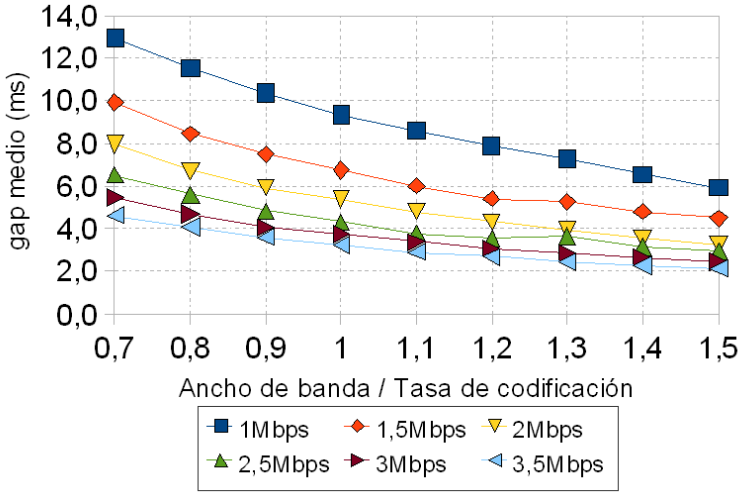


Figura 10.8: Gap para los distintos niveles de ocupación y distintas calidades de los contenidos.

Como se muestra en la figura 10.8, una mejora en el ancho de banda disponible implica una disminución exponencial de los valores de separación entre paquetes consecutivos pertenecientes a un mismo *frame*.

10.1.6.5. Resumen

A modo de resumen podemos concluir que las métricas analizadas presentan una clara dependencia de la disponibilidad de ancho de banda 'q'. Por tanto, parece acertado pensar que podemos emplear estas métricas como indicadores de congestión para obtener la información necesaria para llevar a cabo un proceso de adaptación de la tasa de codificación de los contenidos al ancho de banda disponible en una sesión *streaming*. En el siguiente apartado profundizaremos sobre esta idea.

10.2. Algoritmo de estimación

Ya hemos definido qué parámetros propios de una sesión de *streaming* presentan una relación con la congestión. El siguiente paso será emplear esa información con el objetivo de adaptar la tasa de transmisión de los contenidos multimedia al ancho de banda disponible, utilizando así los recursos de forma eficiente y mejorando la calidad percibida para las condiciones de transmisión dadas. Con la adaptación se pretende dotar de cierta QoS a los procesos de distribución de audio/vídeo a través de Internet, evitando las situaciones que producen pérdida de paquetes y que, como resultado, reducen la calidad causando un impacto negativo en la experiencia de usuario. Cuando se detecte una situación de congestión deberemos disminuir el *bit rate* de transmisión del vídeo. Por el contrario, cuando no haya congestión, el *bit rate* podrá incrementarse.

A continuación explicaremos el diseño del algoritmo de estimación y sus dos variantes propuestas: una primera variante, basada en las métricas de linealidad y gap y que analizaremos en detalle. La segunda propuesta corrige las debilidades que presenta la primera, basándose en las métricas del porcentaje de pérdidas, el *jitter* y la linealidad.

10.2.1. Diseño del algoritmo

La solución propuesta ha sido diseñada sobre protocolos propios para la transmisión de datos multimedia en una sesión de *streaming* (RTP/RTCP) y consiste en emplear los paquetes de *feedback* de RTCP para enviar la información con las métricas analizadas del cliente al servidor. Con esa información el servidor es capaz de identificar los estados de congestión en la red y actuar en consecuencia, respondiendo ante ellas por medio de una codificación adaptativa. Por lo tanto, el sistema puede resumirse en dos fases:

1. Determinar el estado de la transmisión, basándonos en la información de retroalimentación que proviene del cliente.

2. Determinar el valor de la tasa de codificación en función de dicho estado, de manera que consigamos adaptar los contenidos de vídeo al ancho de banda disponible.

A medida que los paquetes de datos RTP llegan al cliente, son procesados y analizados para calcular las métricas vistas en el apartado 10.1. Como hemos explicado, toda la información necesaria para el cálculo de dichas métricas se encuentra en la cabecera de los paquetes RTP (*timestamp*, número de secuencia, instante de recepción y el tamaño del paquete en bytes.). Para aumentar la estabilidad de los valores de las métricas, el cálculo se realizará utilizando la media móvil de los últimos 5 valores.

Una vez calculadas las métricas se envían al servidor por medio de paquetes de control del protocolo RTCP. En concreto, se empleará el tipo de paquetes RTCP-APP ya que se trata de paquetes de uso experimental para el desarrollo de nuevas aplicaciones y es posible personalizar los campos con las necesidades del sistema. Por ejemplo, en nuestro caso se incluiría la información referente a las métricas calculadas.

El envío de estos paquetes se hará de forma periódica, cada 5 segundos. Ésta es una solución de compromiso, ya que un envío muy frecuente de paquetes RTCP-APP podría tener un impacto significativo en la ocupación del canal. Además, los procesos de adaptación de tasa en el servidor, llevados a cabo mediante métodos de transcodificación, serían muy frecuentes e innecesarios en la mayoría de los casos, con el consiguiente consumo de recursos extra en el servidor. Por otro lado, un envío muy poco frecuente podría acarrear que la detección de situaciones de congestión se produjese demasiado tarde.

Destacar en este punto que la sobrecarga producida en la red a causa de la información de *feedback* es insignificante. En los análisis del estimador sobre las pruebas en maqueta efectuados *a posteriori,* podremos calcular ese porcentaje de sobrecarga y veremos que el coeficiente de utilización de ancho de banda apenas se verá incrementado en un 0.014 %. De ahí que podamos clasificar la técnica empleada como no intrusiva.

Una vez que el paquete RTCP-APP llega al servidor, se procesa y con la información que contiene se determina el estado del enlace y la tasa de transmisión de bits óptima.

Este último parámetro se actualiza en el codificador MPEG-4 con el fin de completar el ciclo de estimación. En nuestro caso, aplicaremos un escalado de calidad de manera que se modifican el número de bits por píxel para ajustar la tasa de codificación deseada.

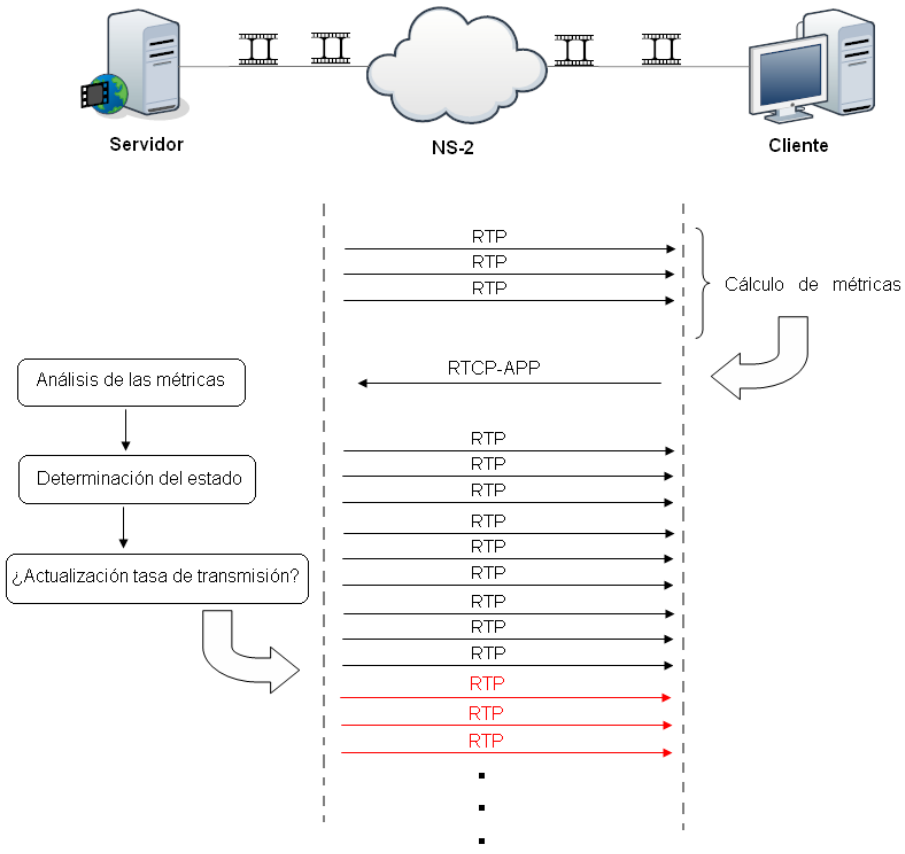


Figura 10.9: Diseño del sistema de adaptación

10.2.2. Estados

En función de los valores de las métricas establecemos una clasificación de los distintos niveles de ocupación de la red:

- Congestión: situación que se produce cuando la tasa de codificación es mayor que el ancho de banda disponible. Relacionándolo con la ecuación 9.3, se correspondería con valores de q<1. Es una situación no deseable, que deberemos corregir en cuanto sea detectada.

- Equilibrio: en este estado no se producirá prácticamente ninguna pérdida de paquetes ya que el ancho de banda disponible es ligeramente superior a la tasa de codificación. Para este caso vamos a establecer que el valor de q se encuentre en el intervalo [1, 1.2). El objetivo del estimador será intentar mantener la sesión en este estado, donde la calidad de los contenidos ofertados es la máxima posible para el ancho de banda disponible.

- Sin congestión: cuando parte del ancho de banda de la conexión está libre. En este caso podemos utilizar el ancho de banda sobrante para mejorar la calidad del vídeo incrementando la tasa de codificación en la medida de lo posible.

El único estado que no requiere ninguna acción sobre los contenidos codificados es el estado de equilibrio. En el resto de los casos habrá que aumentar o disminuir la tasa de transmisión. Pero, ¿cuál es el valor de ese incremento o decremento?

Frente a un estado de congestión es necesario una actuación rápida para adaptarse lo antes posible a las condiciones del canal y evitar las pérdidas y las pausas en la reproducción. Por lo tanto, en cuanto se detecte esta situación se establecerá que la tasa de codificación para la transmisión sea el 90 % de la tasa de recepción que nos indica el cliente (ya que la tasa de recepción en el cliente se corresponderá con la tasa máxima de envío para la situación de congestión). De esta forma evitaremos ajustar los contenidos al límite de la ocupación de la red, disponiendo así de una salvaguarda que evitará que se sobrepase la capacidad en periodos de inestabilidad.

Cuando hay suficiente ancho de banda disponible en la red, deberemos aumentar la calidad de los contenidos. Hay que tener en cuenta que este incremento no debe ser excesivo, porque podríamos inducir una nueva situación de congestión. Por ello se hace necesario estimar los recursos que hay libres y de esta manera conseguiremos que el incremento de la tasa de bits sea acorde a la capacidad sobrante de ancho de banda. El aumento se efectuará de manera progresiva, con incrementos máximos de un 20 % de la tasa de codificación en ese instante. Esta postura conservadora a la hora de realizar los incrementos permite evitar comportamientos de oscilación a la hora de ajustar los contenidos (Rejaie et. al, 1999) . Evitamos de esta manera, por ejemplo, que un aumento excesivo de la tasa produzca que se sobrepase la capacidad del canal, provocando un estado de congestión, con la consiguiente reducción de bit rate en la tasa de codificación y así sucesivamente, hasta finalmente alcanzar una situación estable. Para calcular el incremento que deberemos aplicar a la tasa de codificación nos basaremos en los valores de dos métricas: para la primera versión del algoritmo emplearemos la linealidad y el gap, mientras que para la segunda versión utilizaremos la linealidad y el *jitter*.

10.2.3. Determinación del estado

Ya hemos definido los estados y las actuaciones que hay que llevar a cabo en cada uno de ellos. Pero aún no hemos hablado de cómo se calcula o cómo se determina el estado en que nos encontramos y tampoco se han definido los algoritmos que deberemos ejecutar en cada estado. En la sección 10.2.1 hemos comentado que la determinación del estado se hace en base a las métricas que recibimos del cliente. En esta tesis doctoral vamos a analizar un primer prototipo de estimador basado en métricas como el gap y el coeficiente de correlación de *Pearson*, y un segundo estimador, con ciertas mejoras respecto al primero, basado en el porcentaje de pérdidas, el *jitter* y coeficiente de *Pearson*.

10.2.3.1. Estimador basado en gap y linealidad

Una primera propuesta para el algoritmo de estimación lo constituye el siguiente estimador, basado en las métricas del gap (o separación entre paquetes) y el coeficiente de correlación de Pearson de los instantes de recepción. El valor de estas métricas se calcula

en el cliente, con información contenida en las cabeceras de los paquetes RTP recibidos. Dichas métricas, junto con el valor de la tasa de recepción de los contenidos en el cliente, se comunican al servidor por medio del *feedback* que proporcionan los paquetes RTCP-APP.

La estimación del estado de la transmisión se llevará a cabo en el servidor a través de la métrica de la linealidad, con umbrales establecidos a partir de los análisis empíricos vistos en 10.1.6.3. A cada umbral le corresponde un estado, bien sea el estado de congestión, de equilibrio o sin congestión. Vemos un ejemplo gráfico en la figura 10.10), donde, en función del valor del coeficiente de correlación de Pearson calculado en el cliente, el estado de la transmisión se clasifica en un estado u otro.

Como medida adicional para detectar un estado de congestión, el servidor comprobará también el valor de la tasa de recepción en el cliente. Si la tasa de recepción en el cliente es menor que la tasa a la que se están enviando los contenidos puede ser un indicativo de congestión.

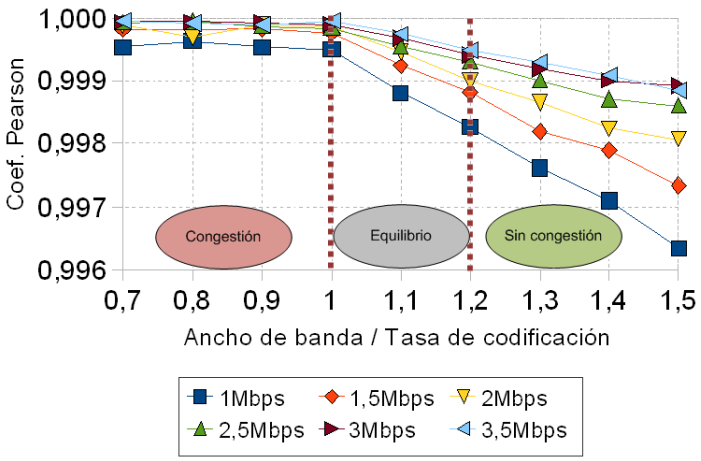


Figura 10.10: Ejemplo de los umbrales establecidos en las métricas para calcular el estado

Las acciones a llevar a cabo, una vez catalogado el estado de la transmisión a través de los umbrales de la métrica de la linealidad y la tasa de recepción en el cliente, son las siguientes:

- Si se detecta congestión en la red por medio de estos parámetros, la nueva tasa de codificación se establecerá con un valor del 90 % de la tasa de recepción que el cliente comunica (que es la tasa a la que está recibiendo los contenidos).

- Si no hay congestión, es necesario calcular el incremento que podemos aplicar a la tasa de transmisión. Dicho incremento se calcula en base al criterio del gap y la linealidad. Por ejemplo, si la métrica de la linealidad presenta un valor muy bajo sería indicativo de un estado sin congestión en el que hay gran cantidad de recursos

libres en la red, por tanto, podríamos elevar notablemente la tasa de codificación de los contenidos que enviamos.

Para cada una de las métricas, se establecen nuevos umbrales empíricos. A cada umbral le corresponde aplicar un incremento en la tasa de codificación. Pero, tal y como se ha especificado en 10.2.2, el porcentaje de subida que se aplique a la tasa de codificación no superará el 20 % para el mejor de los casos.

El resultado final de la estimación es una combinación de ambos criterios, de forma que cada una de las dos métricas aportarán un factor de incremento en la tasa de codificación en función del valor que tomen. Sin embargo, a cada una de las métricas se le puede asignar diferentes pesos para que tengan más o menos influencia sobre el resultado final. De esta manera podemos combinar la influencia que pueda ejercer el criterio del gap o el criterio de la linealidad en la estimación del estado de la transmisión cuando no existe congestión.

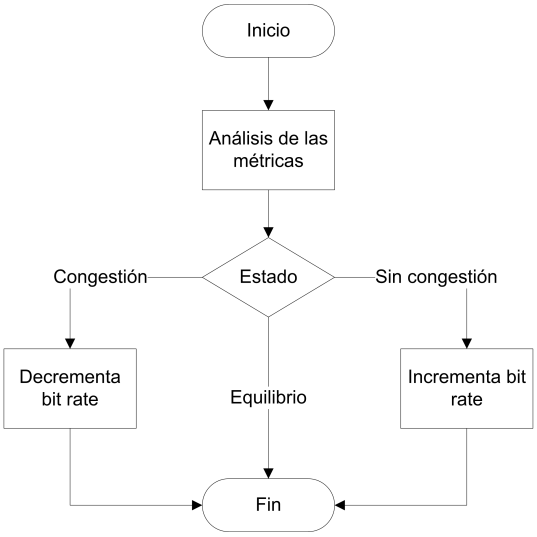


Figura 10.11: Diagrama del algoritmo de adaptación para un estimador basado en las métricas de: gap y linealidad

10.2.3.2. Estimador basado en pérdidas, jitter y linealidad

La pérdida de paquetes es una métrica ampliamente utilizada. Su cálculo es relativamente sencillo gracias al campo *Sequence Number* de la cabecera RTP que se incrementa cada vez que se envía un paquete. De esta manera, el cliente puede detectar cuándo no recibe un paquete. Esta métrica es factible de ser empleada cuando el nivel de ocupación de la red sea del 100 %, momento en el cual se producirá la pérdida de paquetes.

En esta segunda versión del algoritmo, unas pérdidas superiores al 2 % identificarán inmediatamente el estado como congestión. Las acciones a realizar en este caso son análogas a las anteriores.

Si las pérdidas son inferiores al 2 % el siguiente paso será estimar el valor de 'q', definido en la ecuación 9.3. Para ello empleamos la información de las métricas del *jitter* y

la linealidad. En este caso también es posible ajustar la influencia de cada métrica dando más importancia a una u otra en el proceso de adaptación.

Pongamos un ejemplo gráfico de estimación de 'q' a partir de la métrica del *jitter* (ver Figura 10.12). A partir de la información de la gráfica 10.6 podemos calcular rectas de ajuste para cada valor de la tasa de codificación, obteniendo así una expresión matemática del valor del eje X ('q') en función del eje Y (*jitter*, en este ejemplo). En nuestro caso, realizaremos un ajuste exponencial tanto para el *jitter* como para la linealidad. Por tanto, conociendo la tasa de codificación queda fijada la recta de ajuste que deberemos emplear. Junto con el valor del *jitter* es posible extraer el valor de q, tal y como se indica en el ejemplo gráfico 10.12, donde suponemos que la tasa de codificación se sitúa en torno a 1.5Mbps y el valor del *jitter* calculado por el cliente es de unos 9 ms, resultando un valor estimado de q=1.2.

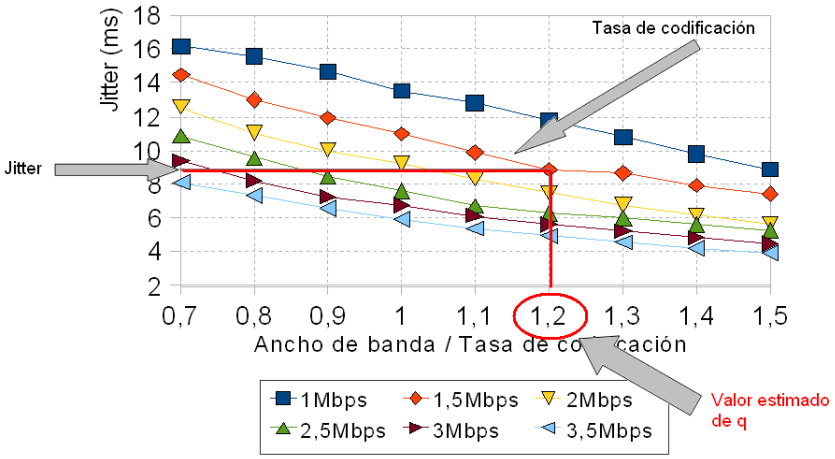


Figura 10.12: Ejemplo del cálculo de q a partir del jitter

Si el resultado estimado de 'q' se encuentra entre valores comprendidos en el intervalo [1, 1.2), estaremos en un estado de equilibrio, donde no será necesario realizar ninguna modificación en la tasa de transmisión. Por el contrario, si el valor de 'q' es mayor de 1.2 se aplicará el incremento en la tasa de codificación en función del valor obtenido.

La figura 10.13 muestra el diagrama de fases para esta versión del algoritmo.

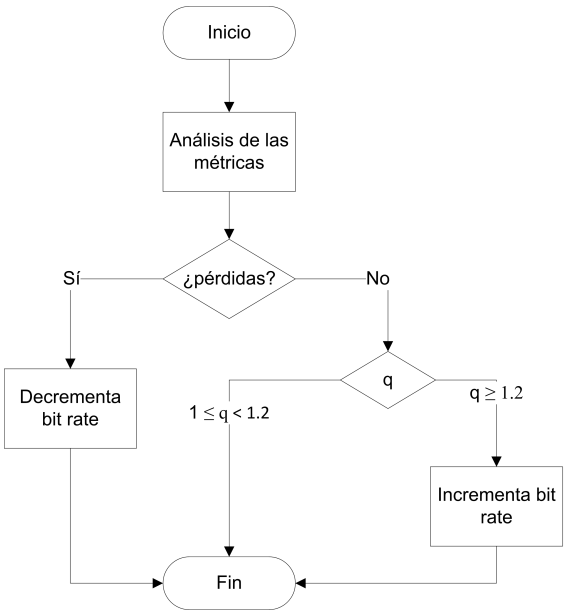


Figura 10.13: Diagrama del algoritmo de adaptación para un estimador basado en las métricas de: pérdidas, jitter y linealidad

10.3. Experimento 2: Evaluación de un estimador basado en GAP y linealidad

Una vez demostrada la necesidad de codificación adaptativa (sección 9.2), y de realizar un estudio en profundidad del tráfico *streaming* (sección 9.2), pasamos a evaluar el primer sistema de estimación propuesto en 10.2.3.1, basando el algoritmo en el criterio del gap y la linealidad de los paquetes en recepción.

10.3.1. Definición de objetivos

Estudiaremos un servicio de vídeo *streaming* adaptativo con contenidos de alta calidad, bajo una arquitectura simple de cliente-servidor con una línea de capacidad variable, desde un punto de vista investigador, con el fin de evaluar el comportamiento del estimador.

El estimador no deberá generar situaciones de oscilación, es decir, el ajuste de la tasa de transmisión (bien sea aumentando o disminuyendo) tiene que hacerse únicamente cuando sea necesario y en el sentido correcto, evitando realizar, por ejemplo, ajustes de incremento de tasa cuando en realidad tenemos una situación con congestión.

10.3.2. Generación de preguntas y definición de métricas

¿Cómo es el comportamiento del sistema adaptativo ante un canal cuya capacidad varía con el tiempo? ¿el algoritmo realiza estimaciones de manera correcta? ¿en situaciones de equilibrio se vuelven a generar ajustes?

El objetivo de un sistema adaptativo es mantener la tasa a la que se sirven los contenidos ajustada al ancho de banda disponible, evitando así que se produzcan pérdidas y otros efectos no deseados que afecten a la calidad de las reproducciones.

Una métrica que indica si el algoritmo ajusta la tasa al ancho de banda disponible es el error cuadrático medio (mse) entre ambas variables (ancho de banda disponible y tasa estimada de transmisión). El mse se define como la media del cuadrado del error, siendo el error la diferencia entre el valor real y el estimado:

$$MSE(\hat{X}) = E\left[(\hat{X} - X)^2\right] \tag{10.7}$$

También deberemos comprobar si se producen sobre-estimaciones y cuál es su amplitud. Las sobre-estimaciones tendrán lugar cuando la tasa de transmisión supere a la capacidad disponible en la red. Sea cual sea la amplitud de la sobre-estimación provocará una situación de pérdida de paquetes y los consiguientes cortes en la reproducción. Por tanto, lo ideal será mantener al mínimo el número de sobre-estimaciones que se produzcan.

10.3.3. Fase de preparación del escenario y de la recogida de datos

El escenario general será similar al empleado en los experimentos anteriores, donde cliente y servidor están conectados a través de una red emulada con NSE. La herramienta de emulación estará ubicada en un tercer equipo y nos permitirá variar en tiempo real las condiciones de transmisión y el ancho de banda disponible. Para este experimento utilizaremos diferentes modelos de variación del canal con el fin de poder evaluar el algoritmo de estimación en diferentes condiciones de red sin necesidad de un despliegue físico, ya que un atributo importante en cualquier algoritmo de control de congestión es su respuesta ante cambios en las condiciones de red.

En el apéndice C podemos ver una explicación detallada de los modelos de tráfico empleados. Utilizamos modelos deterministas y aleatorios, con patrones de tráfico siguiendo una tendencia ascendente o descendente. De esta manera conseguimos caracterizar el comportamiento del estimador en diferentes situaciones. La variación del ancho de banda oscila entre valores de 1Mbps y 5Mbps.

El equipo servidor tendrá instalada una versión modificada del servidor de *streaming* DarwinStreaming Server que incorpora el algoritmo de estimación propuesto, así como la herramienta VLC que permite actualizar y modificar la tasa de codificación de los contenidos.

Por su parte el cliente contará con la versión modificada del VLC que incluye el módulo del cálculo de métricas a partir de la información de los paquetes RTP.

Tanto el cliente como el servidor implementan el sistema de intercambio de información de control basado en paquetes RTCP-APP, tal como se especificó en la descripción del algoritmo de estimación.

El algoritmo de adaptación empleado será el descrito en 10.2.3.1. Para evaluar la mejor configuración de algoritmo, realizaremos pruebas con diferentes pesos para cada una

de las métricas que forman parte del mismo (linealidad y gap). Puesto que el algoritimo depende de dos criterios de ajuste simultáneos, encontrar la solución óptima pasa por evaluar las diferentes combinaciones posibles.

Los datos necesarios para calcular las métricas para la evaluación y análisis del algorimo serán: la capacidad de la red en cada instante y la tasa de codificación. Estos valores se recogerán en tiempo real en ficheros con información extraída directamente de las aplicaciones.

La duración máxima de cada experimento se establecerá en 300 segundos.

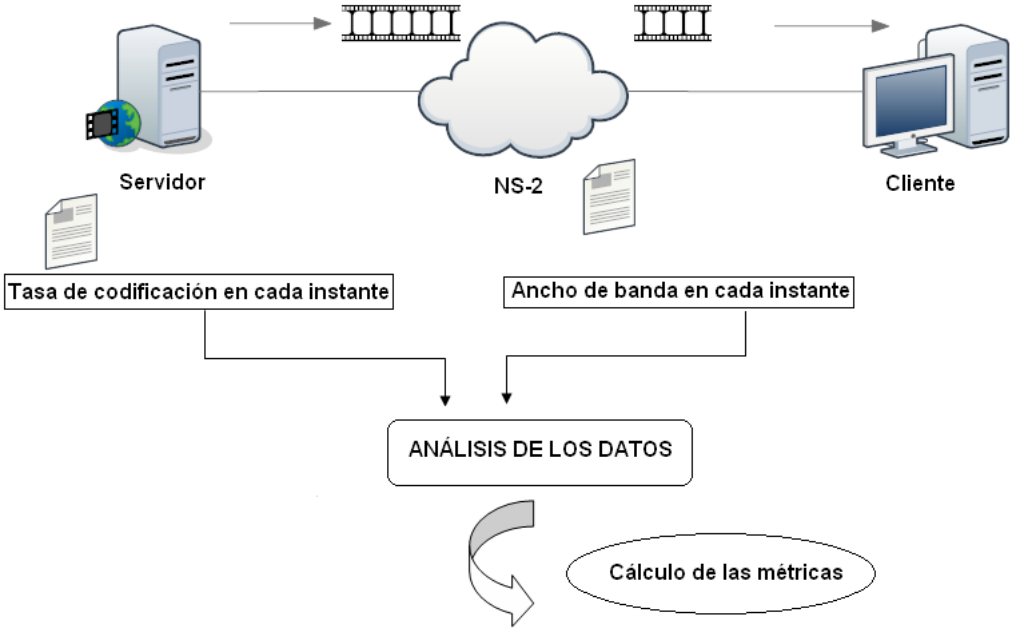


Figura 10.14: Experimento 2 - Evaluación de un estimador basado en GAP y linealidad

Al igual que en experimentos anteriores, es necesario comprobar y validar todo el sistema antes de pasar a la siguiente fase en el desarrollo de la prueba. El resumen de las principales características de esta fase se muestra en la Figura 10.14.

10.3.4. Realización del experimento

Los detalles de la arquitectura y el esquema empleado para llevar a cabo todas las pruebas con los diferentes escenarios pertenecientes a este experimento pueden encontrarse en el apéndice D. Para cada modelo de canal y combinación de criterios se realizarán 50 repeticiones del experimento.

10.3.5. Recogida y validación de los datos

Una vez finalizado el experimento procederemos al análisis de los datos.

10.3.6. Extracción de conclusiones

Como ya hemos comentado, para validar con exactitud el sistema de estimación hemos estudiado su comportamiento frente a diferentes modelos de red. Esta versión del algoritmo emplea como métricas de estimación la linealidad de los paquetes en recepción y el gap. Tomaremos como primer factor el criterio de la linealidad y como segundo factor el criterio del gap, empleando diferentes pesos para cada criterio a la hora de establecer un ajuste óptimo para el algoritmo. En lo sucesivo denotaremos como w_1 al peso que se corresponde con el criterio 1 (linealidad) y el peso w_2 con el segundo criterio (gap). Por ejemplo, la combinación (0.9, 0.1) significa el criterio de la linealidad tendrá una influencia del 90 % en el cálculo del ancho de banda estimado frente al 10 % del criterio del gap.

10.3.6.1. Error cuadrático medio

La siguiente tabla muestra los resultados del error cuadrático medio entre el ancho de banda disponible y los valores estimados.

$w_1 - w_2$	Modelo triangular	Modelo sierra ascendente	Modelo sierra descendente	Modelo tendencia ascendente	Modelo tendencia descendente
1.0 - 0.0	3.40	8.68	12.79	11.37	8.08
0.9 - 0.1	4.04	8.94	9.26	16.10	12.05
0.8 - 0.2	3.65	8.62	7.60	14.28	11.65
0.7 - 0.3	4.30	8.61	8.39	12.63	9.96
0.6 - 0.4	4.46	10.73	9.48	13.84	6.70
0.5 - 0.5	5.81	11.33	10.23	15.07	13.57
0.0 - 1.0	25.07	28.71	14.63	27.74	20.13

Tabla 10.1: Error cuadrático medio

Como podemos apreciar en los valores de la Tabla 10.1, no hemos realizado todo el barrido de pesos, ya que los valores correspondientes a las combinaciones donde se da más importancia al criterio del gap (0.4-0.6; 0.3-0.7; 0.2-0.8; 0.1-0.9; 0.0-1.0) empeoraban el comportamiento del sistema y por tanto no serán combinaciones válidas para un estimador óptimo. Simplemente mostraremos los valores de las métricas para el peor caso (0.0-1.0) a modo de ejemplo.

En la Figura 10.15 vemos una representación del error cuadrático medio para los modelos deterministas del canal (los cuales presentan los mismos valores de ancho de banda en todos las repeticiones del experimento): el modelo triangular, el de sierra ascendente y el de sierra descendente (Sección C.1). Se aprecia visualmente la diferencia entre lo que consideraríamos los mejores casos de configuración del algoritmo (de 1.0-0.0 a 0.5-0.5) frente al peor de los casos (0.0-1.0), que presenta un mse notablemente mayor para todas las configuraciones de variación del canal.

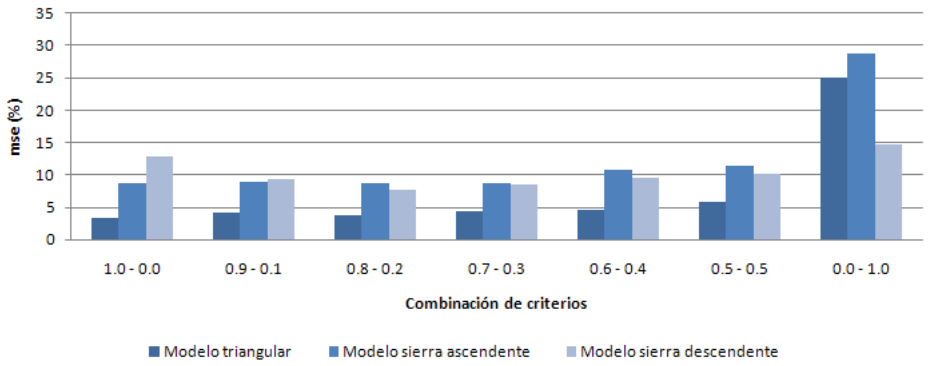


Figura 10.15: Error cuadrático medio

10.3.6.2. Sobreestimaciones

Otra métrica necesaria para evaluar el estimador es el porcentaje de sobreestimaciones que tienen lugar y su amplitud. La Tabla 10.2 resume los valores obtenidos en la evaluación de este parámetro.

$w_1 - w_2$	Modelo triangular	Modelo sierra ascendente	Modelo sierra descendente	Modelo tendencia ascendente	Modelo tendencia descendente
1.0 - 0.0	19.31	4.76	30.73	13.55	24.23
0.9 - 0.1	16.27	5.05	17.14	8.69	21.39
0.8 - 0.2	17.01	6.44	22.14	11.02	17.25
0.7 - 0.3	16.19	5.48	25.24	8.97	15.74
0.6 - 0.4	16.62	6.98	22.40	9.75	18.04/
0.5 - 0.5	17.75	6.64	22.21	8.72	17.36
0.0 - 1.0	7.77	6.26	16.41	7.62	5.70

Tabla 10.2: Porcentaje total de sobreestimaciones

Observamos un elevado porcentaje de sobreestimaciones en los modelos de canal que cuentan con pendientes descendientes en su configuración (ver Figura 10.16). En en apartado 10.3.6.3 explicaremos el por qué de estos valores con ayuda de un ejemplo gráfico.

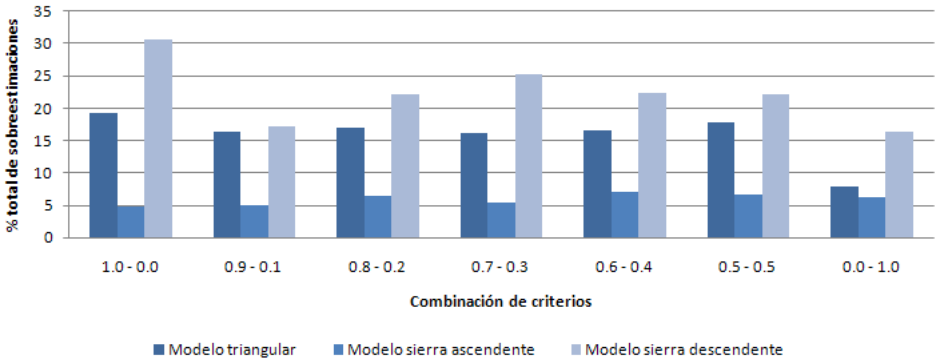


Figura 10.16: Porcentaje total de sobreestimaciones

$w_1 - w_2$	Modelo triangular	Modelo sierra ascendente	Modelo sierra descendente	Modelo tendencia ascendente	Modelo tendencia descendente
1.0 - 0.0	3.46	< 1	4.95	< 1	3.04
0.9 - 0.1	5.28	< 1	3.12	< 1	2.79
0.8 - 0.2	3.15	< 1	3.93	< 1	1.80
0.7 - 0.3	2.01	< 1	3.70	< 1	4.84
0.6 - 0.4	3.85	< 1	4.03	< 1	1.53
0.5 - 0.5	3.84	< 1	4.02	< 1	2.07
0.0 - 1.0	4.59	< 1	4.40	< 1	3.64

Tabla 10.3: Porcentaje de la amplitud de las sobreestimaciones

En general, la amplitud de las sobreestimaciones es reducida (Tabla 10.3). Sin embargo, en algunas configuraciones de red se produce un elevado número de las mismas, lo que dará lugar a situaciones de congestión y pérdidas, sea cual sea la amplitud de dichas sobreestimaciones.

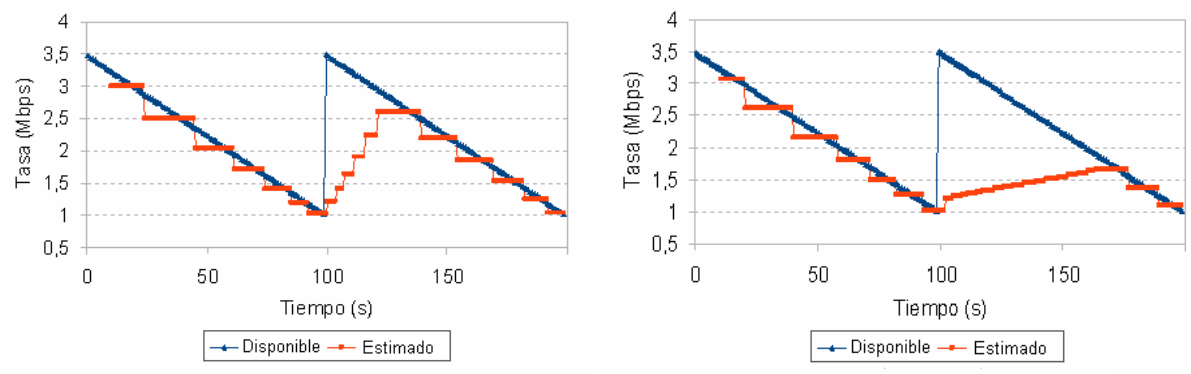
Analizando los resultados de ambas tablas, podemos establecer que las soluciones óptimas se encontrarían en el rango de combinaciones de (0.7 , 0.3) - (0.9 , 0.1), en función del modelo de capacidad de red.

10.3.6.3. Algunos ejemplos gráficos

A la vista de los resultados obtenidos, la combinación que produciría menor error cuadrático medio y menor número de sobreestimaciones para los diferentes modelos deterministas del canal tiene lugar con la combinación (0.8, 0.2).

A continuación veremos un análisis que muestra el impacto de cada métrica del algoritmo (linealidad y gap) a la hora de calcular los porcentajes de subida en la tasa de codificación, poniendo de manifiesto la importancia de ajustar de manera correcta el peso de cada criterio. Veremos que se obtienen menores errores si damos más peso al criterio de la linealidad. A modo de ejemplo, mostraremos el proceso de adaptación de la combinación de criteros que consideramos óptima (0.8, 0.2) frente al peor caso de ajuste (cuando

solo entra en juego el criterio del gap, (0.0, 1.0)).

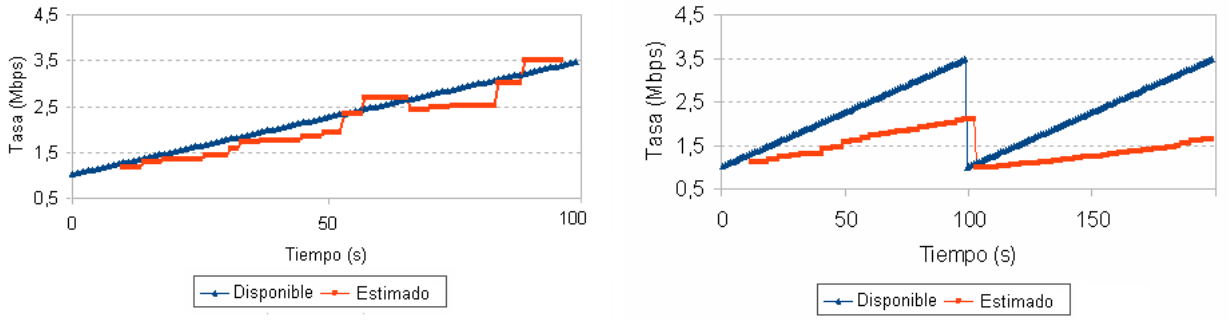


(a) Mejor caso (b) Peor caso

Figura 10.17: Ejemplo del mejor caso vs el peor caso de adaptación para un modelo de canal de sierra descendente

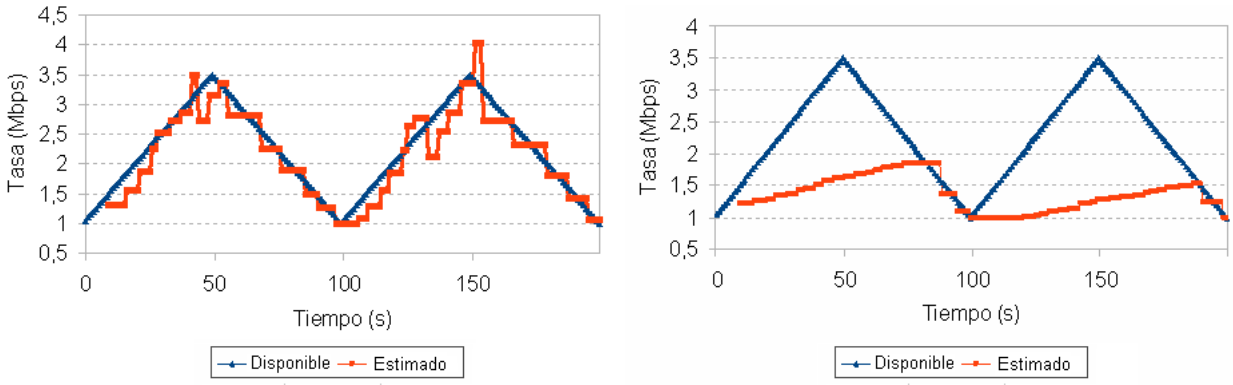
El peor caso lo presenta cuando anulamos la linealidad y sólo tenemos en cuenta el criterio del gap. Esto es debido a que el criterio del gap es mucho más restrictivo y los porcentajes de subida que se aplican en la tasa de transmisión, teniendo en cuenta únicamente este criterio, son mucho menores. De hecho, se ha observado que esta métrica sólo es efectiva cuando las variaciones en el ancho de banda son muy acusadas. En esos casos especiales sí se aprecian variaciones significativas en la separación media entre paquetes (gap), y por tanto se aplica un porcentaje de incremento de tasa de codificación más elevado. Para pequeñas variaciones de ancho de banda, la variación del gap resulta inapreciable, y por consiguiente, el incremento que se aplica en la tasa de codificación es mínimo. Esto hace que la adaptación a las subidas en esta configuración sea muy lenta, provocando un aumento en el error cuadrático medio en la estimación. Vemos un ejemplo claro en la figura 10.17b, donde mostramos el peor caso: en el instante 100, momento en el cual hay un aumento brusco de la capacidad, el ancho de banda estimado no sigue la misma progresión de incremento que el ancho de banda disponible. El escenario homólogo (Figura 10.17a), pero con una combinación óptima, realiza incrementos consecutivos del valor máximo establecido (un 20 %) hasta alcanzar más rápidamente la situación de estabilidad. El mismo fenómeno descrito se aprecia en la Figura 10.18b, donde podemos ver un caso donde los incrementos de tasa estimada que se aplican son mínimos, con lo que la adaptación en modelos de canal ascendente es lenta. Para visualizar este hecho correctamente mostramos un eje de tiempos mayor que para su homólogo correspondiente al mejor caso.

En las siguientes figuras (Figuras 10.18 y 10.19) vemos más ejemplos donde apreciamos gráficamente el proceso de adaptación al ancho de banda disponible en la red. En algunos casos vemos que se producen sobreestimaciones, sobre todo en procesos de bajada y algunas incluso en los procesos de subida. Esta situación no es deseable. Por ejemplo, en la Figura 10.18a y 10.19a se observan periodos en los que el estimador calcula una tasa de transmisión superior al ancho de banda que está disponible en la red durante un proceso de incremento de ancho de banda.



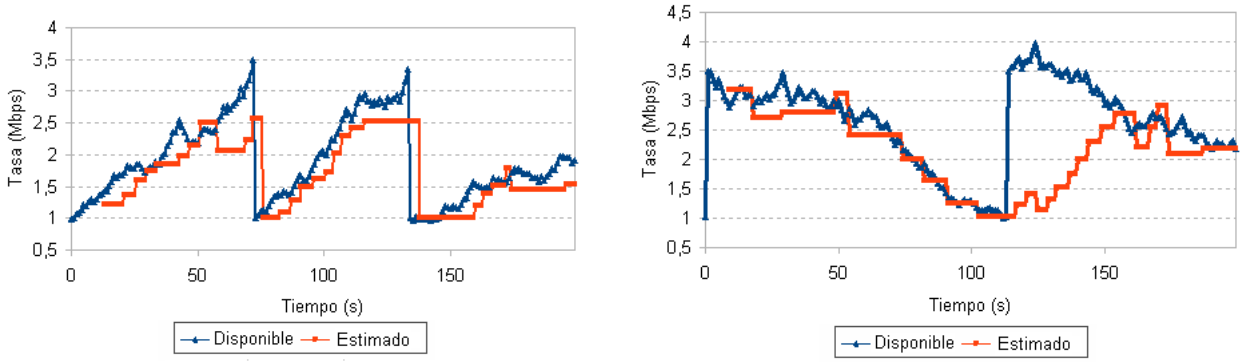
(a) Mejor caso (b) Peor caso

Figura 10.18: Ejemplo del mejor caso vs el peor caso de adaptación para un modelo de canal de sierra ascendente



(a) Mejor caso (b) Peor caso

Figura 10.19: Ejemplo del mejor caso vs el peor caso de adaptación para un modelo de canal triangular



(a) Tendencia ascendente (b) Tendencia descendente

Figura 10.20: Ejemplos para canales con tendencia ascendente y tendencia descendente

Las sobreestimaciones durante los procesos de bajada de la capacidad del canal tienen su explicación en que la actualización de la tasa se realiza cada 5 segundos. Si durante esos intervalos el ancho de banda sigue disminuyendo, se producirán periodos de sobreestimación hasta que la tasa vuelva a ser ajustada. Por tanto, en los modelos con canal descendente, las sobreestimaciones se producirán durante breves periodos de tiempo (un máximo de 5 segundos) hasta que el estimador se ajuste y descienda la tasa de codificación (con una salvaguarda del 10 % en la bajada).

Las sobreestimaciones durante los instantes en los que la capacidad de la red está aumentando serán motivo para concluir que esta elección de parámetros de estimación de ancho de banda no parece totalmente acertada, y nos llevará a buscar otras soluciones o alternativas como, por ejemplo, el segundo algoritmo propuesto en la sección 10.2.3.2. Además, el ajuste se hace dejando muy poco margen libre de recursos, lo que provocará pérdidas cuando la situación de red sea inestable. Además, en este algoritmo no se tiene en cuenta la tasa de transmisión y en el estudio realizado en el experimento 1, en la sección 10.1, vimos que las métricas dependían de ella.

Por todo lo dicho anteriormente, buscaremos otra propuesta de implementación para el algoritmo de estimación, ya que consideramos que el error cuadrático medio y el porcentaje de sobreestimaciones que tienen lugar es elevado. Esto nos lleva a pensar que los parámetros de estimación no están bien elegidos y, además, en este sistema las decisiones se toman en base a datos discretos, con lo cual pueden ser poco precisos. A continuación, se evaluará un estimador basado en pérdidas, jitter y linealidad con el fin de encontrar resultados estables que los obtenidos con el presente algoritmo durante el proceso de estimación y adaptación.

10.4. Experimento 3: Evaluación de un estimador basado en pérdidas, jitter y linealidad

En ciertas situaciones, los resultados del estimador basado en GAP y *Pearson* no son todo lo buenos que cabría esperar. Por tanto, tendremos que analizar otros parámetros en el algoritmo.

10.4.1. Definición de objetivos

Igual que para el caso 10.3.1

10.4.2. Generación de preguntas y definición de métricas

Visto en 10.3.2

10.4.3. Fase de preparación del escenario y de la recogida de datos

Presentamos el mismo escenario que en 10.3.3, modificando únicamente el algoritmo empleado. En este caso la configuración será la descrita en 10.2.3.2.

10.4.4. Realización del experimento

Ver detalles en el apéndice D. Al igual que en el experimento anterior, las realizaciones se repetirán 50 veces con el fin de obtener resultados estables.

10.4.5. Recogida y validación de los datos

Se analizan los datos capturados para proceder al cálculo de las métricas propuestas.

10.4.6. Extracción de conclusiones

Al igual que en el apartado 10.3.6, emplearemos diferentes pesos a la hora de ajustar las métricas que forman parte del algoritmo. En este caso será necesario ajustar el *jitter* y la linealidad, ya que las pérdidas, en cuanto se detecten, se declarará una situación de congestión. Únicamente el *jitter* y la linealidad toman partido en los incrementos.

En este caso: w_1 será el criterio de la linealidad y w_2 será el criterio del *jitter*.

Como veremos en los resultados, la mejora en los resultados respecto a la configuración del experimento 3 es claramente apreciable.

10.4.6.1. Error cuadrático medio

Para este algoritmo, es posible realizar pruebas y análisis con todas las combinaciones posibles de criterios. Si comparamos los resultados obtenidos en la Tabla 10.4 (representada gráficamente en la Figura 10.21 para los escenarios deterministas) con la Tabla 10.1 vemos que, para la mayoría de los casos, el error cuadrático medio se reduce.

$w_1 - w_2$	Modelo triangular	Modelo sierra ascendente	Modelo sierra descendente	Modelo tendencia ascendente	Modelo tendencia descendente
1.0 - 0.0	4.27	5.45	3.07	9.55	10.94
0.9 - 0.1	4.63	6.18	2.9	9.51	8.06
0.8 - 0.2	4.91	5.75	2.96	9.04	9.14
0.7 - 0.3	4.24	5.56	2.76	9.26	8.67
0.6 - 0.4	3.72	5.54	2.86	10.16	9.35
0.5 - 0.5	3.66	5.47	3.03	8.77	7.87
0.4 - 0.6	3.88	5.31	2.55	9.46	7.81
0.3 - 0.7	4.01	5.57	2.27	8.05	8.24
0.2 - 0.8	3.99	5.35	2.12	8.34	7.33
0.1 - 0.9	4.19	5.81	2.2	9.66	7.49
0.0 - 1.0	4.37	5.98	1.91	9.96	8.2

Tabla 10.4: Error cuadrático medio

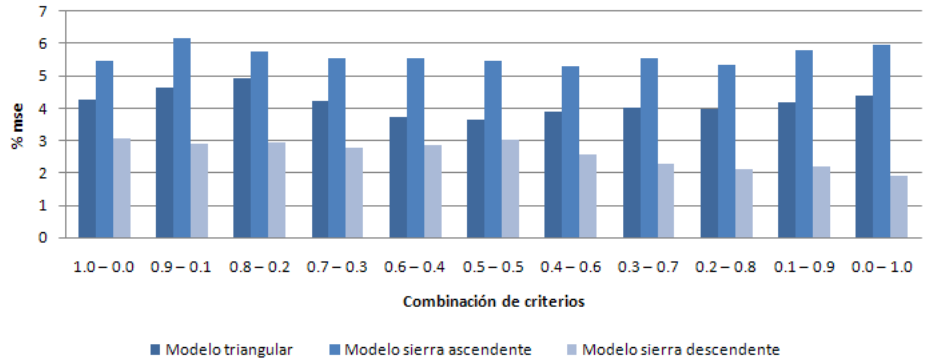


Figura 10.21: Error cuadrático medio

A diferencia de los resultados de la Tabla 10.1, en este caso las diferentes configuraciones de los criterios no presentan valores tan dispares. Parece que para un mismo modelo de canal no resulta muy relevante el peso que se de a uno u otro criterio. Esto puede ser indicativo de que las métricas seleccionadas son las correctas, ya que ninguna parece influenciar la estimación con valores erróneos, como ocurría en el experimento 3 (apartado 10.3).

Puesto que el comportamiento parece depender principalmente del modelo de red, pasamos a analizar dos ejemplos de una misma variación de canal pero con distinta pendiente. Concretamente, el primero de los modelos presenta una pendiente con un valor absoluto la mitad que el segundo modelo, tanto para un caso ascendente como descendente. La tabla 10.5 muestras los resultados numéricos de estos experimentos con diferentes pendientes y la figura 10.22 su representación gráfica.

$w_1 - w_2$	Modelo ascendente 1	Modelo ascendente 2	Modelo descendente 1	Modelo descendente 2
1.0 - 0.0	5.45	6.69	3.07	1.75
0.9 - 0.1	6.18	7.62	2.9	1.49
0.8 - 0.2	5.75	7.84	2.96	1.84
0.7 - 0.3	5.56	6.92	2.76	1.48
0.6 - 0.4	5.54	5.9	2.86	1.48
0.5 - 0.5	5.47	5.89	3.03	1.36
0.4 - 0.6	5.31	6.31	2.55	1.37
0.3 - 0.7	5.57	6.56	2.27	1.38
0.2 - 0.8	5.35	6.61	2.12	1.28
0.1 - 0.9	5.81	7.13	2.2	1.23
0.0 - 1.0	5.98	7.55	1.91	1.01

Tabla 10.5: Error cuadrático medio en función de la pendiente

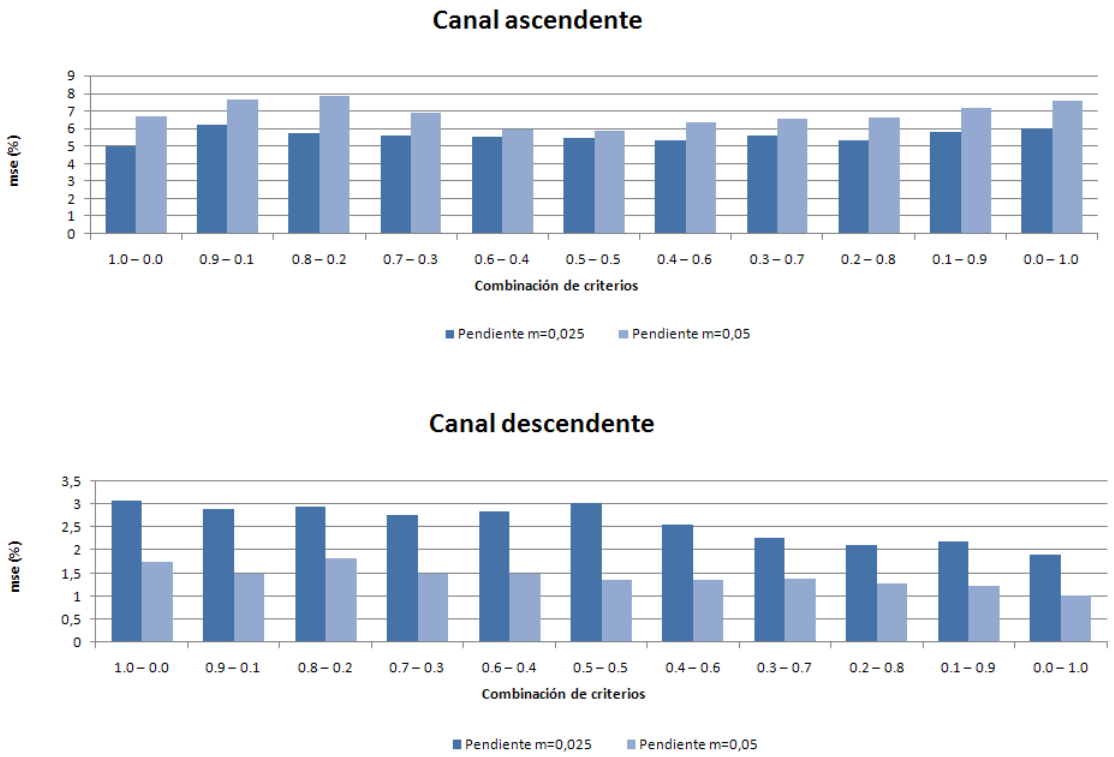
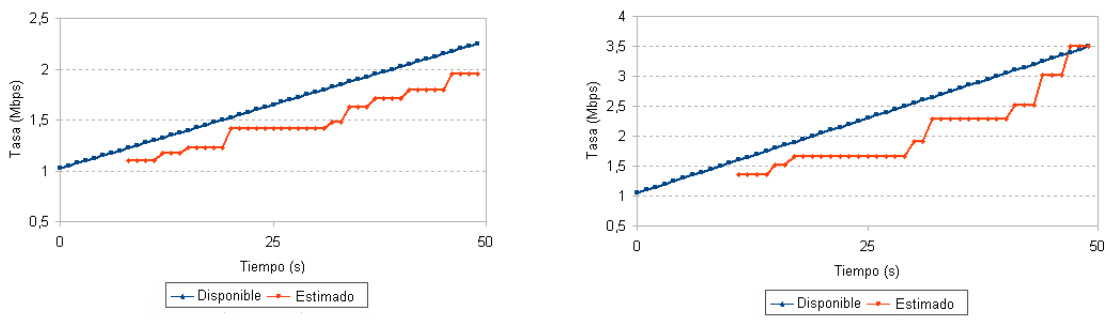


Figura 10.22: mse en función de la pendiente

Para el caso del modelo ascendente, la configuración que presenta más pendiente tienen un mayor mse. Esto es así porque los incrementos que se producen en el ancho de banda disponible son mayores. Sin embargo, recordemos que los incrementos máximos permitidos por el algoritmo son del 20 %, con lo que el proceso de adaptación no podrá seguir el ritmo de subida del canal, produciéndose una diferencia mayor entre el ancho de banda disponible y el estimado.

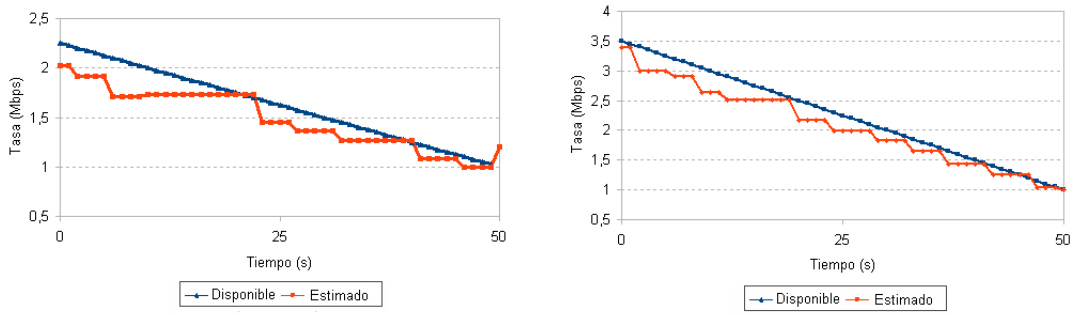
En la Figura 10.23 podemos ver el ejemplo gráfico del proceso de adaptación para un escenario con diferentes pendientes durante el incremento del ancho de banda disponible. Sin embargo, las diferencias no son excesivas y no se aprecia apenas.



(a) Pendiente1 (b) Pendiente2

Figura 10.23: Ejemplos del canal ascendente con diferentes pendientes

Este concepto cambia cuando la tendencia del canal es descendente (Figura 10.24), donde los casos en los que la bajada es más pronunciada se traducen en menor error cuadrático medio.



(a) Pendiente1 (b) Pendiente 2

Figura 10.24: Ejemplos del canal descendente con diferentes pendientes

10.4.6.2. Sobreestimaciones

La tabla 10.6 y la figura 10.25 se corresponden con los resultados referentes al porcentaje total de sobreestimaciones. Estos resultados mejoran claramente a los resultados del estimador basado en gap y linealidad. Este hecho es una ventaja clara de este estimador frente al anterior, ya que la reducción del porcentaje de sobreestimaciones implica que se reduce el porcentaje de pérdidas en el vídeo en recepción.

$w_1 - w_2$	Modelo triangular	Modelo sierra ascendente	Modelo sierra descendente	Modelo tendencia ascendente	Modelo tendencia descendente
1.0 - 0.0	4.24	7.66	10.65	5.97	12.82
0.9 - 0.1	4.72	9.78	9.69	6.69	12.05
0.8 - 0.2	3.92	4.62	8.06	5.01	9.74
0.7 - 0.3	4.12	4.51	8.81	4.58	11.61
0.6 - 0.4	3.7	3.65	8.49	4.17	8.98
0.5 - 0.5	4.19	2.9	8.66	3.44	10.74
0.4 - 0.6	4.06	1.98	11.19	2.47	14.04
0.3 - 0.7	5	1.62	18.75	1.52	14.31
0.2 - 0.8	4.36	0.52	24.72	1.13	16.05
0.1 - 0.9	5.45	0.2	25.37	1.45	16.84
0.0 - 1.0	5.4	0.32	27.31	2.03	18.35

Tabla 10.6: % total de sobreestimaciones

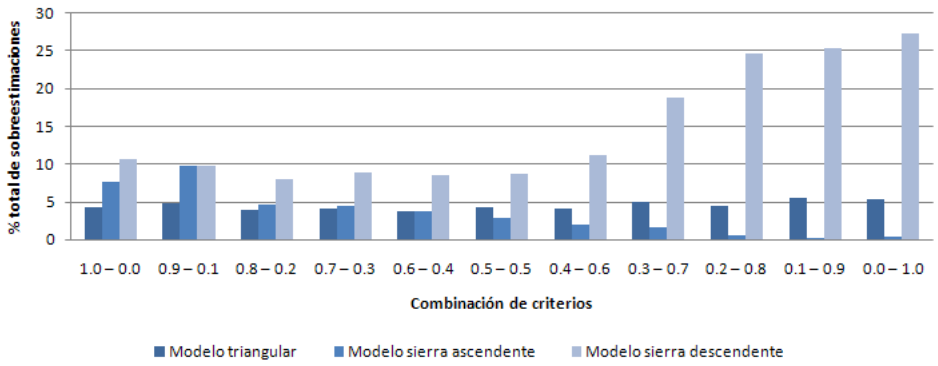


Figura 10.25: % total de sobreestimaciones

Esta mejora es debida a que, gracias al *jitter*, podemos prevenir situaciones de congestión antes de que se produzcan. En los ejemplos gráficos de la sección 10.4.6.3 lo veremos.

Por otro lado, la Figura 10.25 arroja que, para el análisis de las sobreestimaciones, la combinación de los criterios sí tiene importancia para algunos modelos de canal. Concretamente, los modelos con pendientes descendentes en la variación del ancho de banda se ven influenciados negativamente por el criterio del jitter.

De la misma manera que en el análisis del error cuadrático medio (Sección 10.4.6.1), mostramos las variaciones que pueden sufrir los procesos de estimación en cuanto al análisis de las sobreestimaciones en función de las pendientes de subida o bajada del canal (Tabla 10.7 y Figura 10.26). Para esta métrica, cuanto mayor sea la pendiente, menor serán el número de sobreestimaciones que se produzcan.

$w_1 - w_2$	Modelo ascendente 1	Modelo ascendente 2	Modelo descendente 1	Modelo descendente 2
1.0 - 0.0	7.66	2.19	10.65	6.28
0.9 - 0.1	9.78	2.08	9.69	7.35
0.8 - 0.2	4.62	1.44	8.06	6.39
0.7 - 0.3	4.51	1.25	8.81	6.99
0.6 - 0.4	3.65	1.43	8.49	5.97
0.5 - 0.5	2.9	1.51	8.66	6.87
0.4 - 0.6	1.98	1.21	11.19	6.91
0.3 - 0.7	1.62	0.51	18.75	9.49
0.2 - 0.8	0.52	0.2	24.72	8.51
0.1 - 0.9	0.2	0.05	25.37	10.85
0.0 - 1.0	0.32	0	27.31	10.79

Tabla 10.7: % total de sobreestimaciones en función de la pendiente

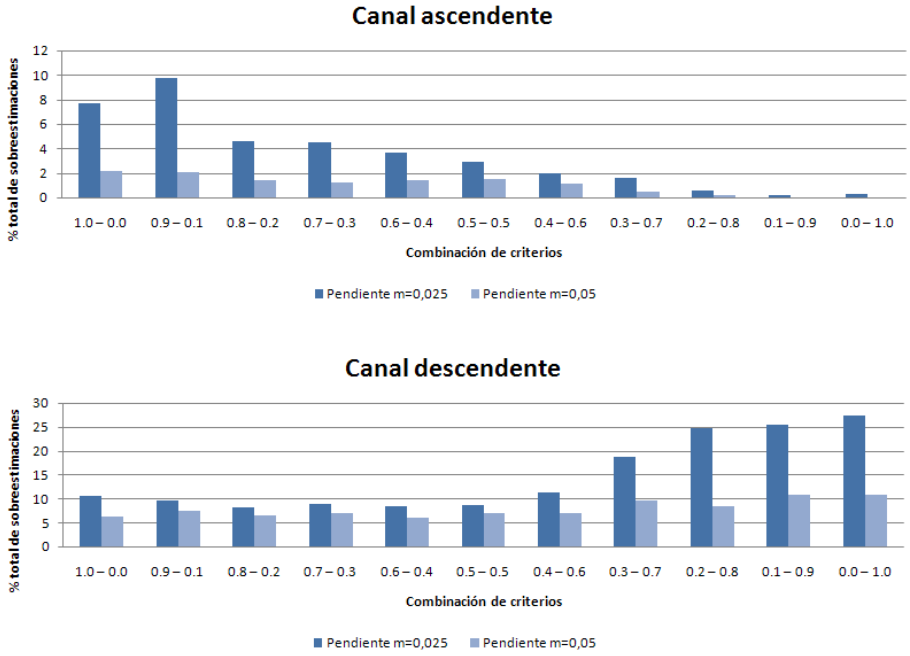


Figura 10.26: % total de sobreestimaciones en función de la pendiente

En cuando a la amplitud de las sobreestimaciones podemos concluir para este caso que los resultados empeoran con respecto a los obtenidos en 10.3.6.2. Sin embargo, aún podemos considerar que la amplitud no es muy elevada, ya que en el peor de los casos presenta un valor en torno al 7 %.

$w_1 - w_2$	Modelo triangular	Modelo sierra ascendente	Modelo sierra descendente	Modelo tendencia ascendente	Modelo tendencia descendente
1.0 - 0.0	3.45	3.9	6.05	4.56	7.39
0.9 - 0.1	4.69	3.13	6.13	3.72	5.91
0.8 - 0.2	3.09	2.93	4.07	2.74	5.84
0.7 - 0.3	3.5	2.5	3.622	2.86	6.69
0.6 - 0.4	3.13	1.98	2.38	2.55	5.31
0.5 - 0.5	2.91	1.33	2.27	2.06	5.37
0.4 - 0.6	2.78	<1	2.67	1.59	5.77
0.3 - 0.7	2.79	<1	3.33	<1	6.5
0.2 - 0.8	2.49	<1	4.02	<1	5.79
0.1 - 0.9	2.93	<1	4.34	<1	5.37
0.0 - 1.0	2.69	<1	4.24	1.24	5.62

Tabla 10.8: % amplitud sobreestimaciones

10.4.6.3. Algunos ejemplos gráficos

Tras analizar los resultados de las métricas que evalúan el comportamiento del estimador, pasamos a ver algunos ejemplos gráficos del proceso de adaptación.

En la Figura 10.27 vemos un ejemplo donde apenas se producen sobreestimaciones. En general, en las pruebas realizadas con modelos ascendentes las sobreestimaciones son prácticamente nulas gracias a una mejor elección y ajuste de las métricas.

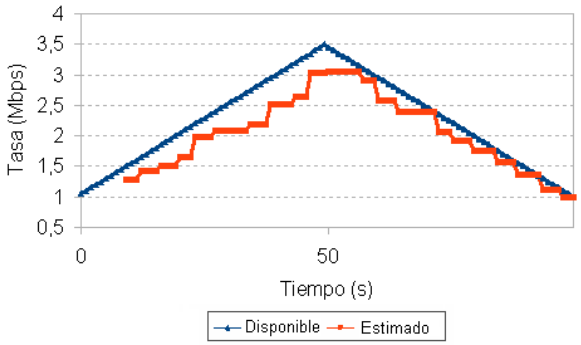
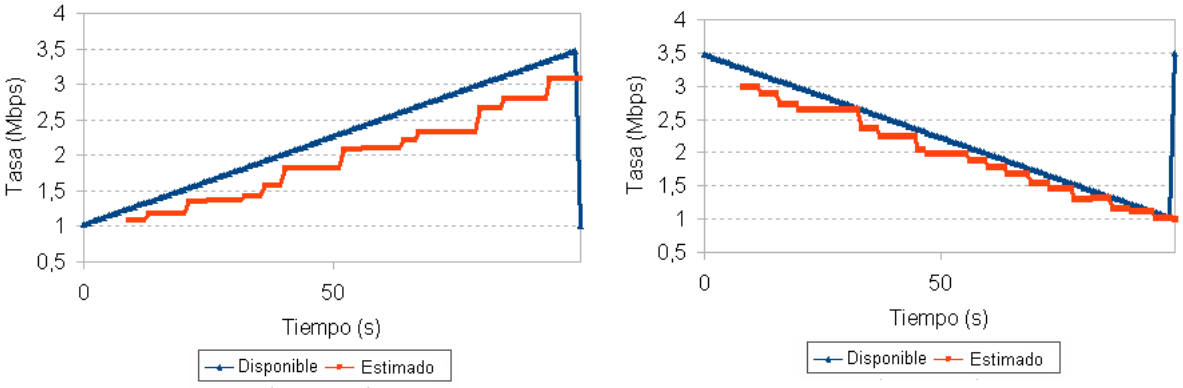


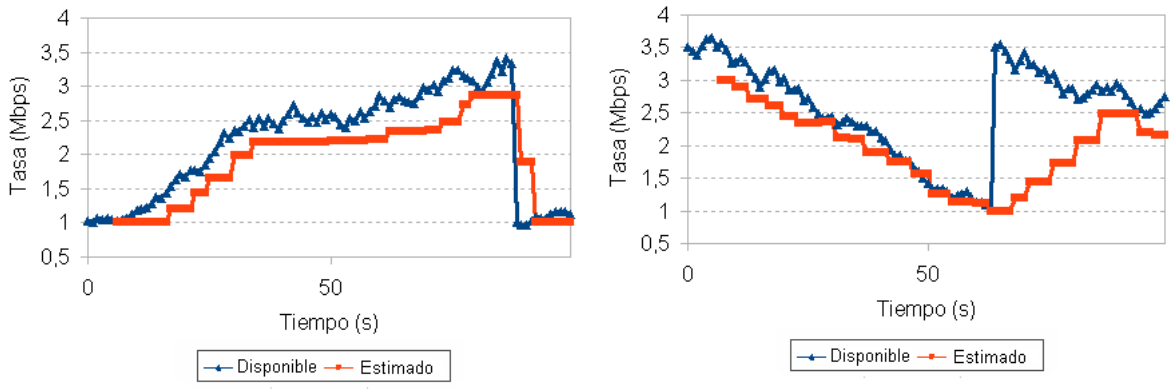
Figura 10.27: Ejemplo del modelo triangular

En las Figura 10.28b y 10.29b encontramos la mayor diferencia con respecto al algoritmo del experimento visto en la sección 10.3 (experimento 2), ya que en este caso, con las métricas elegidas para los procesos de estimación y adaptación es posible anticiparse a las situaciones de congestión cuando la capacidad de la red disminuye, evitando los procesos de pérdida de paquetes y sobreestimaciones que veíamos en 10.17a.



(a) Sierra ascendente (b) Sierra descendente

Figura 10.28: Ejemplo para los modelos de sierra



(a) Tendencia ascendente (b) Tendencia descendente

Figura 10.29: Ejemplo para los modelos aleatorios

Como podemos observar, en modelos de capacidad de canal creciente, Figuras 10.28ay 10.29a, se van produciendo ajustes sucesivos, buscando llegar a una situación de equilibrio. Si durante este proceso hay un descenso en la capacidad, se invierte la tendencia (Figura 10.27), y viceversa: si hay congestión y se están produciendo decrementos en la tasa y en un momento dado aumenta la capacidad de la red, se invierte la tendencia.

Además cuando se alcanza la situación de equilibrio el estimador no realiza nuevos ajustes.

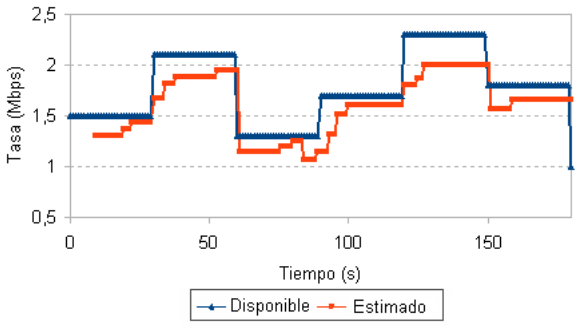


Figura 10.30: Ejemplo del modelo escalón

10.4.6.4. Comparación entre los algoritmos

Con el algoritmo basado en pérdidas, jitter y linealidad conseguimos estabilizar al error cuadrático medio, obteniendo valores menores en la mayoría de los casos (Figura 10.31).

Por otro lado, el porcentaje total de sobreestimaciones se reduce notablemente, tal y como muestra el gráfico de la Figura 10.32. Este hecho es muy significativo ya que las sobreestimaciones provocan pérdidas de paquetes y estas situaciones se tienen que evitar

en la medida de lo posible. Especial relevancia tiene la disminución de este parámetro en modelos de canal descendente, donde las sobreestimaciones con el algoritmo explicado en la sección 10.2.3.1 serán frecuentes. En el nuevo sistema, la tasa estimada de transmisión se sitúa ligeramente por debajo del ancho de banda disponible, evitando realizar una estimación muy ajustada a los recursos libres, lo que provocaría pérdidas en situaciones de inestabilidad de ancho de banda. Esto es posible gracias a que con el *jitter* podemos predecir futuras situaciones de congestión. Esta característica también es importante para los modelos de canal de capacidad creciente, donde conseguimos evitar los picos de sobreestimaciones que veíamos en algunos ejemplos de la sección 10.3.6.3.

Para el caso de la amplitud de las sobreestimaciones, para alguna de las configuraciones del canal, su valor es ligeramente mayor que en el experimento 3. Sin embargo, en el peor de los casos, dicha amplitud no supera el 7 %, y recordemos que el número total de sobreestimaciones que se producen se redujo considerablemente, por lo que la amplitud de las mismas no se traduce en un peor funcionamiento del sistema.

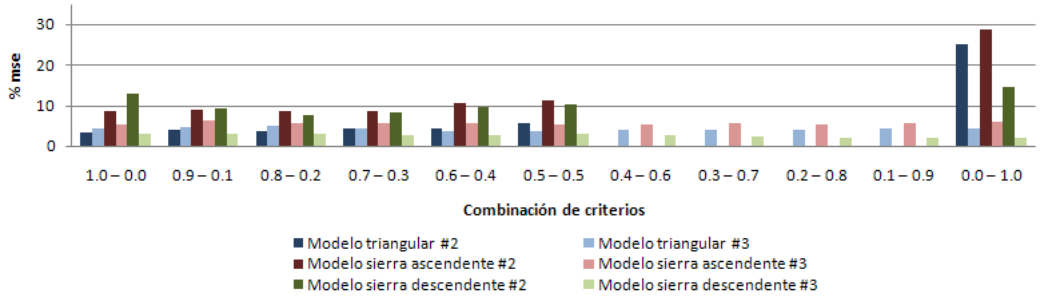


Figura 10.31: Comparación del mse para el experimento 2 y 3

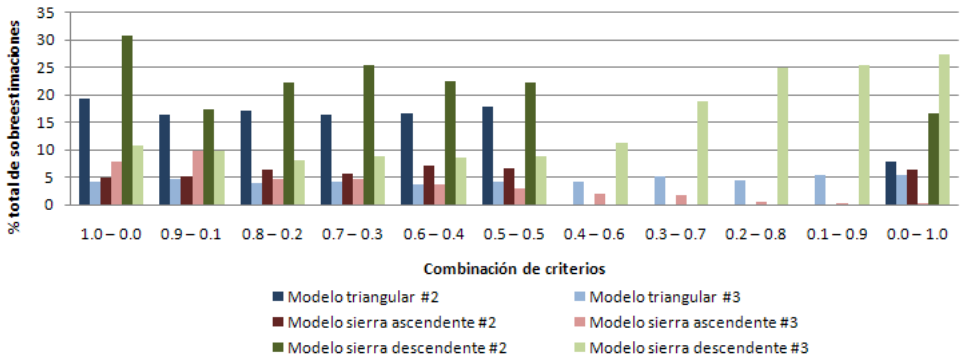


Figura 10.32: Comparación del % total de sobreestimaciones para el experimento 2 y 3

10.4.6.5. Tiempos de adaptación

Por último, aunque no estaba contemplado en las métricas de evaluación del estimador, vamos a analizar los tiempos de respuesta. Este análisis extra lo realizamos a modo de completar el estudio del estimador basado en pérdidas, *jitter* y linealidad, ya que los resultados que se obtienen con esta configuración son más estables y mejores, por lo que,

en los sucesivos experimentos, será el algoritmo que empleemos.

El periodo de adaptación depende del tipo de variación en el ancho de banda: si el ancho de banda disminuye, el periodo de adaptación es corto (entre 2 y 4 segundos), ya que la disminución de la tasa se realiza de manera inmediata. Por el contrario, si el ancho de banda sufre un aumento brusco, el periodo de adaptación será mucho mayor (llegando incluso a los 14 segundos) ya que, según el diseño del algoritmo, el aumento de tasa se realiza de manera progresiva, de manera que es posible que sean necesarios varios ajustes sucesivos hasta alcanzar la tasa objetivo. La Figura 10.33 muestra los resultados de los tiempos de adaptación para la configuración de pesos establecida como óptima (0.8-0.2).

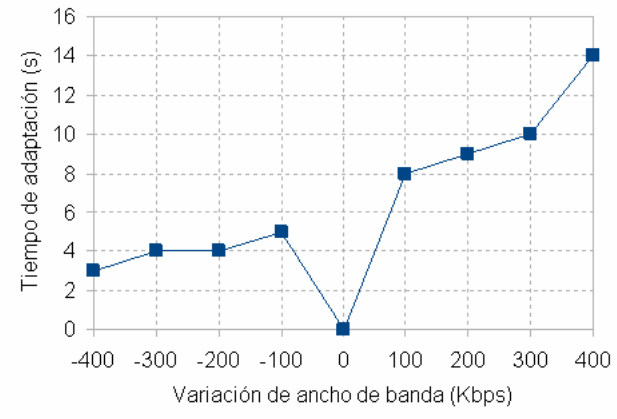


Figura 10.33: Tiempos de adaptación

10.5. Experimento 4: Evaluación del estimador con presencia de otro tráfico

Una vez definidos correctamente los parámetros del algoritmo, pasamos a comprobar el comportamiento del sistema y analizar el rendimiento cuando está presente otro tráfico en la red, ya que será muy común encontrarnos con esta situación de competencia de recursos.

Cabe destacar que en las líneas de acceso del usuario no se aplica ninguna política de QoS en función del tráfico, esto es, no importa si se trata de tráfico http, ftp o *streaming*. La única excepción a esta regla la presenta el tráfico de VoIP, que sí recibe un tratamiento prioritario sobre el resto de los servicios.

Por lo tanto estudiaremos cómo convive el tráfico *streaming* del sistema adaptativo con otro tráfico presente en la red.

10.5.1. Definición de objetivos

Estudiaremos, desde un punto de vista investigador, los servicios de vídeo *streaming* adaptativos con contenidos de alta calidad, bajo una arquitectura simple de cliente-servidor en un entorno cableado, cuando en la red está presente otro tipo de tráfico.

10.5.2. Generación de preguntas y definición de métricas

¿Cuál es el comportamiento del sistema adaptativo de *streaming* cuando en la red hay un consumo de recursos por parte de otro tŕafico? Para un correcto funcionamiento, la suma de los anchos de banda consumidos por las diferentes conexiones no deberán superar el ancho de banda total del enlace. Esto se relaciona directamente con las sobreestimaciones que se produzcan. Nuestro algoritmo deberá reaccionar ante estos casos.

Por tanto, la métrica que evalúa el objetivo propuesto es el ancho de banda total consumido por la sesión de *streaming* y el tráfico adicional.

10.5.3. Fase de preparación del escenario y de recogida de datos

Puesto que nuestro objetivo en este experimento será evaluar el estimador en presencia de otro tráfico, vamos a mantener constante el ancho de banda del enlace. Por otro lado habrá que considerar qué tipo de servicio va a competir con la sesión *streaming*. Hemos optado por tráfico CBR (*Constant Bit Rate*), donde podemos controlar el ancho de banda consumido.

Lo que haremos será introducir tráfico CBR en diferentes instantes para comprobar que el sistema de adaptación identifica la congestión producida por la presencia de ese tráfico.

Con NS-2 podemos fijar y conocer la capacidad del enlace en cada instante (aquí será constante) y volcaremos esa información a un fichero, junto con la información de los instantes donde se emula tráfico CBR y su correspondiente valor.

Por otro lado, en otro fichero almacenaremos la tasa a la que se están sirviendo los contenidos.

Para validar el escenario (Figura 10.34), en este caso habrá que incluir una comprobación del comportamiento del tráfico CBR, revisando que el ancho de banda que consume es el especificado en los ficheros de configuración.

10.5.4. Realización del experimento

Visto en otros apartados similares de experimentos anteriores.

10.5.5. Recogida y validación de los datos

Visto en otros apartados similares de experimentos anteriores.

10.5.6. Extracción de conclusiones

Como vemos en la Figura 10.35, entre los segundos 50 y 100 se ha inyectado tráfico cuya tasa es de 500Kbps en un canal cuya capacidad total de ancho de banda es de 2.5Mbps. Algo similar ocurre entre los instantes 200 y 230, donde el tráfico presenta un valor de 250Kbps. Durante estos intervalos, el sistema es capaz de calcular el ancho de

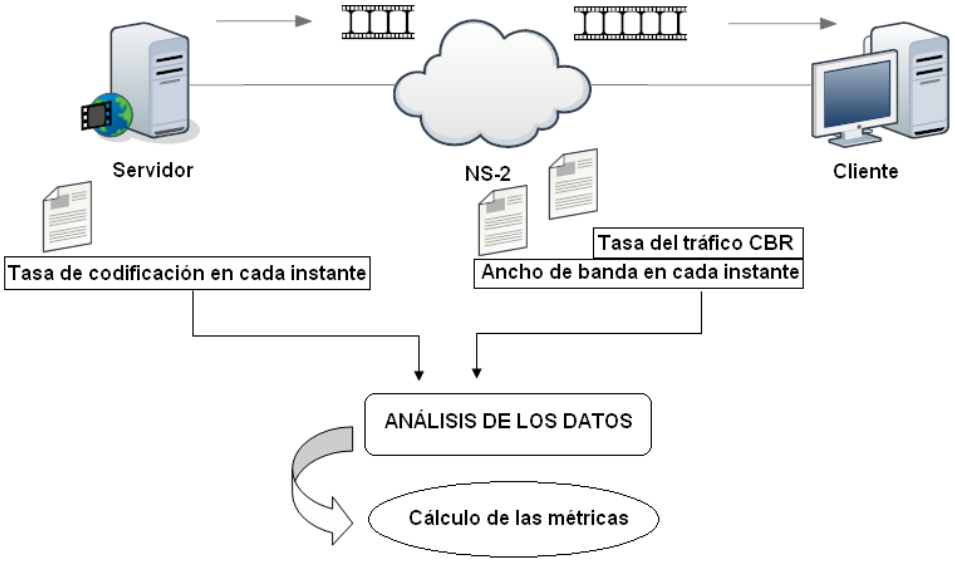


Figura 10.34: Experimento 4

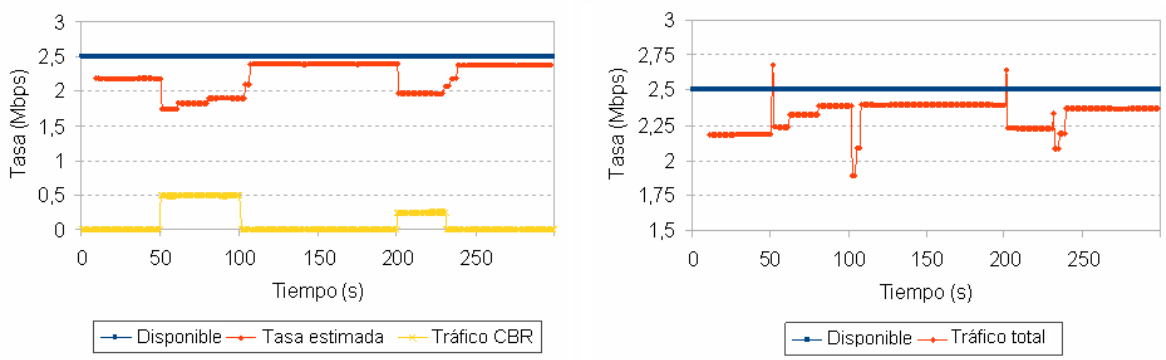


Figura 10.35: Ejemplo con tráfico de fondo

banda que queda libre, teniendo en cuenta ese tráfico adicional. Por consiguiente, para el primer intervalo, por ejemplo, el ancho de banda disponible estimado se encuentra en torno a 2Mbps.

Comentar también que en este caso apreciamos sobreestimaciones en los momentos iniciales en los que el tráfico *streaming* y el tráfico CBR conviven (instantes 50 y 200). Esto es debido a que la recepción de paquetes con las métricas necesarias para calcular la congestión tiene lugar cada 5 segundos. Con lo cual, el valor estimado se mantendrá hasta la próxima adaptación, independientemente del aumento repentino de la congestión. A continuación, durante el siguiente periodo de estimación de la congestión, se resuelve la situación actualizando la tasa de bits a la nueva situación detectada.

10.6. Experimento 5: Análisis de escalabilidad del estimador

El sistema de estimación y adaptación propuesto cumple el objetivo principal de llevar a cabo una correcta adaptación a las condiciones variables de ancho de banda. Pero, ¿qué pasa con la escalabilidad del sistema?

En el diseño original del sistema de estimación contempla la opción de que varios clientes se conecten al servicio, de manera que, para cada uno de ellos, se establece una sesión en la que la tasa de transmisión del vídeo se ajusta al canal de cada cliente en particular. La limitación de este caso es la gran cantidad de recursos que se necesitan en el servidor, ya que para cada cliente se inicia un proceso de transcodificación para adaptar los contenidos en tiempo real.

10.6.1. Definición de objetivos

Estudiaremos servicios de vídeo *streaming* adaptativos con contenidos de alta calidad, bajo una arquitectura simple de cliente-servidor en un entorno cableado, desde un punto de vista de investigación con el propósito de evaluar la escalabilidad del sistema.

10.6.2. Generación de preguntas y definición de métricas

¿Cuántos usuarios es capaz de soportar el servicio?

La métrica para evaluar la escalabilidad será el número de usuarios concurrentes.

10.6.3. Fase de preparación del escenario y de la recogida de datos

Cuando el usuario accede al servicio de *streaming* adaptativo está consumiendo un porcentaje importante de recursos en la máquina del servidor. La causa de este fenómeno tiene que ver con los procesos de transcodificación. Para un correcto funcionamiento del sistema, el porcentaje de uso de CPU deberá mantenerse por debajo de los valores críticos aceptados (en torno al 80 %).

De la misma forma, en el cliente también analizaremos y comprobaremos el porcentaje de uso de CPU.

Para ello emplearemos la herramienta SAR[3]. SAR es una herramienta que está incluida en el paquete Sysstat para Linux. El objetivo de estas herramientas es la monitorización de parámetros de rendimiento. En concreto, la herramienta SAR permite monitorizar información de la actividad del sistema, tales como CPU, memoria, interrupciones, interfaces de red, etc.

En este punto resultará interesante destacar las características físicas de los equipos empleados. Podemos encontrar los detalles en el apéndice B en la sección B.1.

[3]http://linuxcommand.org/man_pages/sar1.html

Para este experimento, representado en la Figura 10.36, contemplaremos diferente número de clientes accediendo al servicio, cada uno de ellos con unas características determinadas en su red de acceso, con el fin de emular clientes procedentes de distintas áreas geográficas.

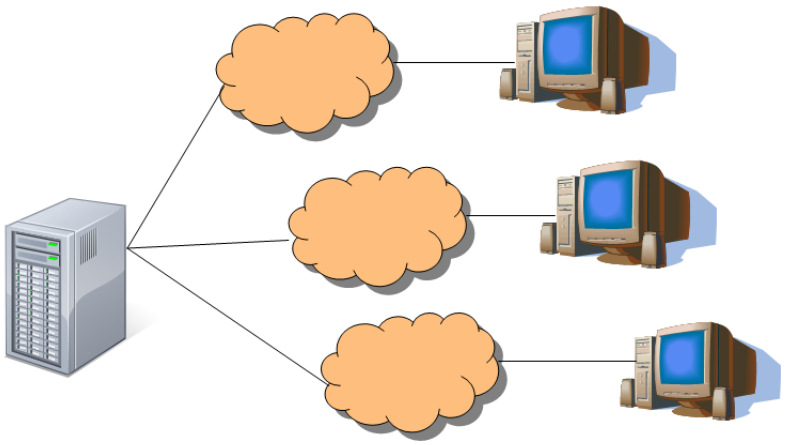


Figura 10.36: Experimento 5

10.6.4. Realización del experimento

Se ejecutan las realizaciones del experimento en el escenario descrito. En este caso, los clientes acceden al servicio de manera escalonada.

10.6.5. Recogida y validación de los datos

Visto en otros apartados similares de experimentos anteriores.

10.6.6. Extracción de conclusiones

Analizando los resultados del uso de CPU en el cliente podemos concluir que, el incremento de uso de CPU en el cliente adaptativo con respecto a un cliente no adaptativo es ínfimo, estando por debajo del 0.5 %. Sin embargo, el servidor se ve rápidamente sobrecargado a medida que el número de clientes aumenta. De hecho, el número máximo de clientes que soporta esta máquina, manteniendo el límite superior de uso de CPU por debajo del 80 %, es de 3 usuarios simultáneos.

En la Tabla 10.9 se resume el porcentaje de uso de CPU en el servidor cuando hay 1, 2 ó 3 clientes accediendo al servicio de manera simultánea. Un cuarto cliente ya no podría acceder al servicio, ya que el servidor se encuentra altamente cargado con los procesos de transcodificación del resto de clientes.

	1 cliente	2 clientes	3 clientes
% uso de CPU en el servidor	39.73	58.79	66.282

Tabla 10.9: % de uso de CPU en función del número de clientes

Podemos comparar los datos anteriores con los datos de CPU de un servidor que no realiza tareas de adaptación de contenidos, observando la clara diferencia de consumo de recursos entre un sistema adaptativo y un sistema no adaptativo (Tabla 10.10)

	1 cliente	2 clientes	3 clientes
% uso de CPU en el servidor	11.49	12.99	13.41

Tabla 10.10: % uso de CPU para un servidor no adaptativo

A continuación analizamos el proceso de adaptación con clientes simultáneos. Como vemos en la Figura 10.37, el cliente 1 presenta un canal con una capacidad del enlace que sigue una tendencia ascendente, el ancho de banda disponible para el cliente 2 sigue una tendencia descendente y el cliente 3 presenta un modelo triangular. En estas condiciones tan dispares para cada cliente, el sistema es capaz de llevar a cabo una correcta adaptación de los contenidos para cada uno de ellos, enviando el *stream* de vídeo correspondiente a cada cliente, con la tasa de transmisión adecuada.

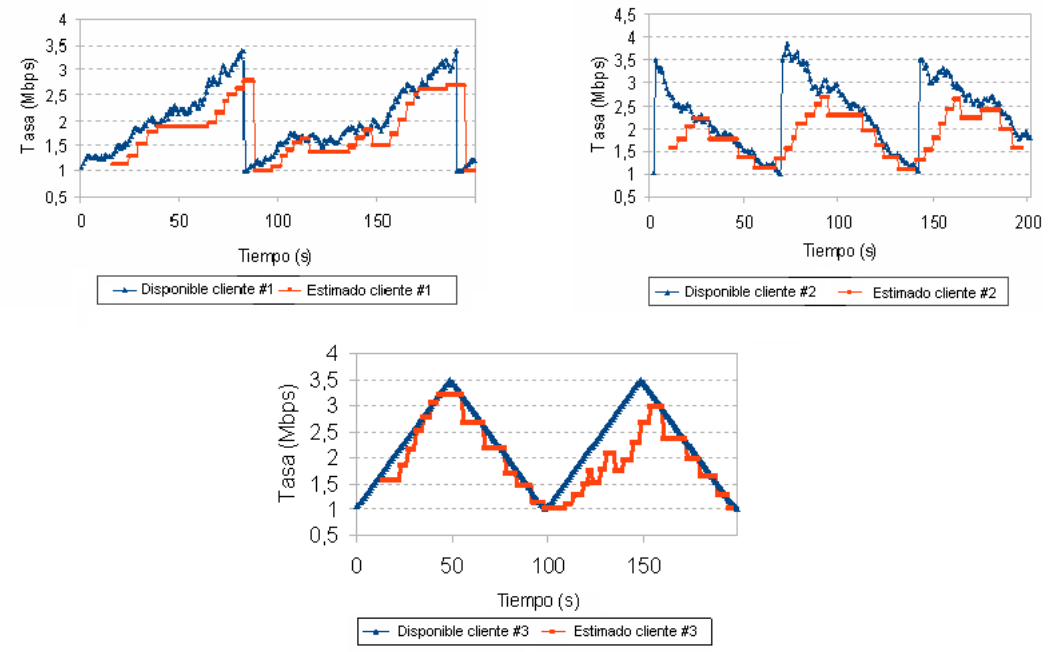


Figura 10.37: Resultados de escalabilidad

Concluimos por tanto que, el principal inconveniente de esta arquitectura es el elevado consumo de CPU que se produce en el servidor, a consecuencia de los procesos de transcodificación. Cuando el servidor recibe peticiones de nuevos clientes se produce una notable bajada del rendimiento. Sin embargo, cuando el número de clientes está dentro de los límites aceptados, el servidor realiza de manera correcta los procesos de adaptación. Por tanto, si buscamos un sistema altamente escalable no podremos recurrir a arquitecturas donde la técnica de adaptación consista en transcodificar los contenidos. Tendremos que buscar otras alternativas, como podría ser SVC (Schwarz et. al, 2007).

10.7. Experimento 6: Evaluación de la QoE del estimador basado en pérdidas, jitter y linealidad

En la sección 9.2 correspondiente al experimento 1 veíamos el análisis de la experiencia de usuario cuando no se dispone de un sistema adaptativo para ajustar la tasa de envío de los contenidos. En este experimento realizaremos la misma evaluación, pero empleando una arquitectura de distribución de vídeo por Internet que cuenta con el sistema de estimación y adaptación visto en las secciones anteriores.

10.7.1. Definición de objetivos

Estudiaremos el servicio de vídeo *streaming* adaptativo, bajo una arquitectura simple de cliente-servidor en un entorno cableado, desde un punto de vista de investigación.

El objetivo será la evaluación de la calidad de las reproducciones en el cliente bajo distintas condiciones de ocupación en la red.

10.7.2. Generación de preguntas y definición de métricas

Visto en el apartado 9.2.2.

10.7.3. Fase de preparación del escenario y de la recogida de datos

El escenario planteado será similar al del experimento 1 (visto en el apartado 9.2.3), donde se establecía una sesión *streaming* entre el cliente y el servidor y se almacenaban el vídeo original y el recibido para el posterior cálculo de las métricas. La única diferencia con este escenario se encuentra en el tipo de sesión establecida, ya que, para este caso, contaremos con el sistema de estimación propuesto en el experimento 3 (apartado 10.4).

De la misma forma, los modelos de variación de canal para esta evaluación serán los mismos que los mencionados en los experimentos anteriores, y que se encuentran definidos en detalle en el Apéndice C, Sección C.1.

10.7.4. Realización del experimento

Visto en otros apartados similares de experimentos anteriores.

10.7.5. Recogida y validación de los datos

Visto en otros apartados similares de experimentos anteriores.

10.7.6. Realización del experimento

Visto en otros apartados similares de experimentos anteriores.

10.7.7. Recogida y validación de los datos

Visto en otros apartados similares de experimentos anteriores.

$w_1 - w_2$	Modelo triangular	Modelo sierra ascendente	Modelo sierra descendente	Modelo tendencia ascendente	Modelo tendencia descendente
1.0 - 0.0	27.49	30	29.08	29.46	28.61
0.9 - 0.1	26.84	29.78	29.56	29.15	28.84
0.8 - 0.2	27.86	30.23	29.32	29.68	28.6
0.7 - 0.3	27.29	30.02	28.99	29.36	28.59
0.6 - 0.4	27.53	29.85	29.17	29.15	28.55
0.5 - 0.5	27.34	29.71	29.26	29.06	28.88
0.4 - 0.6	27.18	30.01	28.14	29.34	28.03
0.3 - 0.7	26.9	30.25	28.62	29.66	28.37
0.2 - 0.8	27	30.16	28.51	29.81	27.82
0.1 - 0.9	26.22	29.96	28.2	29.11	28.3
0.0 - 1.0	26.78	29.87	28.01	28.49	27.72

Tabla 10.11: PSNR

10.7.8. Extracción de conclusiones

Los resultados en la sección 9.2 mostraban valores de QoE (*Quality of Experience*) muy por debajo de un resultado satisfactorio en cuanto se producían pérdidas a causa de una disminución de ancho de banda. A continuación se presentan los resultados de la experiencia de usuario de un sistema de distribución de vídeo adaptativo. Concretamente, el algoritmo empleado será el descrito en 10.2.3.2.

En la tabla 10.11 vemos los resultados para la métrica de la PSNR en todas las configuraciones posibles del algoritmo y con diferentes modelos de capacidad de canal. Prácticamente la totalidad de los valores se encuentran en el intervalo 3 de la escala MOS que califica la reproducción como aceptable.

Veamos ahora, en la Figura 10.38, el porcentaje de *frames* que se corresponden a cada intervalo de la escala MOS (representados en la leyenda en la escala 1 (peor) a 5 (mejor)) para un modelo de canal con una tendencia aleatoria descendente, a modo de ejemplo para un caso en el que la red de acceso del usuario esté cada vez más congestionada y por tanto con mayores problemas durante la reproducción.

Observamos que, en prácticamente todas las configuraciones, cerca del 80 % de los *frames* obtienen una puntuación excelente en la escala MOS. Esta situación es muy diferente a la vista en 9.5, donde, en los casos en los que la red presentaba congestión, la calidad de las reproducciones era muy pobre. Con el sistema de adaptación evitamos estas situaciones tan indeseables, mejorando sustancialmente la experiencia de usuario.

De la misma manera, los valores de SSIM para este caso son excelentes, presentando valores próximos a la unidad (Tabla 10.12).

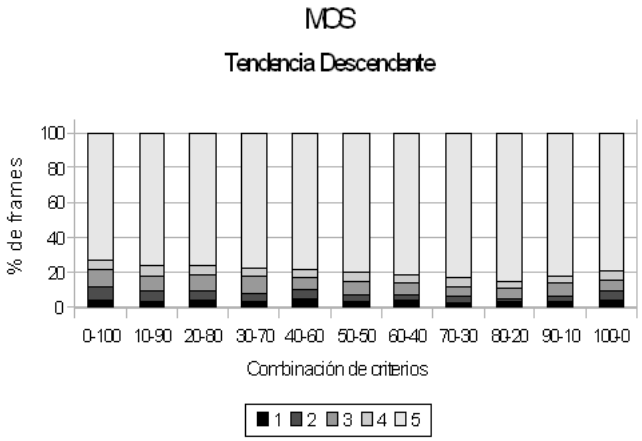


Figura 10.38: MOS

$w_1 - w_2$	Modelo triangular	Modelo sierra ascendente	Modelo sierra descendente	Modelo tendencia ascendente	Modelo tendencia descendente
1.0 - 0.0	0.958	0.989	0.976	0.986	0.965
0.9 - 0.1	0.951	0.989	0.978	0.984	0.968
0.8 - 0.2	0.959	0.990	0.977	0.987	0.970
0.7 - 0.3	0.955	0.989	0.977	0.986	0.968
0.6 - 0.4	0.959	0.990	0.976	0.983	0.967
0.5 - 0.5	0.955	0.987	0.977	0.986	0.968
0.4 - 0.6	0.954	0.990	0.958	0.987	0.958
0.3 - 0.7	0.950	0.991	0.964	0.988	0.962
0.2 - 0.8	0.950	0.991	0.959	0.989	0.955
0.1 - 0.9	0.940	0.990	0.956	0.987	0.958
0.0 - 1.0	0.945	0.986	0.951	0.965	0.951

Tabla 10.12: SSIM

Por tanto, la valoración objetiva del sistema *streaming* adaptativo sitúa la experiencia de usuario en rangos de aceptación elevados. Este factor será la clave para el éxito del servicio.

10.8. Conclusiones finales

Este capítulo resume el diseño e implementación de sistemas de vídeo *streaming* adaptativos basados en técnicas de transcodificación.

El análisis del tráfico *streaming* en un sistema tradicional cliente-servidor ha desvelado la posibilidad de incluir nuevas métricas para su uso en procesos no intrusivos de estimación del ancho de banda disponible. Concluímos así que el análisis de los instantes de llegada de los paquetes en el receptor ha demostrado ser un claro indicativo de congestión.

La construcción de los algoritmos de estimación y adaptación se realiza basado en protocolos *streaming* tradicionales, esto es, RTP y RTCP, facilitando así la integración en un entorno real.

Tras el estudio de diferentes propuestas, el estimador basado en el porcentaje de paquetes perdidos, el *jitter* y la linealidad, muestra los mejores resultados en cuanto a su comportamiento frente a diferentes modelos de variación del canal.

La principal ventaja que nos aportan las técnicas de transcodificación es la posibilidad de abarcar un rango continuo de valores en el bitrate de codificación. Sin embargo, la tecnología se ve fuertemente limitada en el aspecto de escalabilidad del servicio, ya que el número de usuarios concurrentes que puede manejar es reducido.

Capítulo 11

ESTIMACIÓN Y ADAPTACIÓN EMPLEANDO CODIFICADORES ESCALABLES

Según lo visto en el capítulo anterior, las técnicas basadas en trascodificación presentan el inconveniente de la sobrecarga de recursos en el servidor. Por tanto, la escalabilidad de esta solución constituye su principal desventaja. Pero existen otros mecanismos que permiten que las aplicaciones *streaming* sean soportadas en redes *best-effort*. Una posibilidad la brinda SVC (*Scalable Video Coding*), una nueva tecnología que permite codificar un fichero de vídeo en diferentes calidades dentro del mismo *bitstream*. Esto incluye diferentes resoluciones, diferentes frecuencias (fps) y diferentes calidades, (bits por píxel). Es decir, SVC permite escalabilidad espacial, temporal y de calidad. Este concepto permite la adaptación del vídeo transmitido al ancho de banda disponible o a los parámetros del servicio. En caso de congestiones en la red, donde el ancho de banda no es suficiente para transmitir un vídeo con la máxima calidad, podemos reducir la resolución, los *frame*s por segundo o la calidad de la imagen y, por tanto, el ancho de banda necesario, con el fin de evitar la pérdida de paquetes y una mayor degradación de la calidad de la experiencia (QoE) del usuario. La combinación de la tecnología SVC con algoritmos de estimación de la congestión permite construir un sistema adaptativo que es capaz de adecuar los contenidos a las condiciones de red en cada momento.

Este capítulo aborda el diseño, implementación y evaluación de un servicio de *streaming* adaptativo, empleando las ideas que se plantearon en el capítulo anterior, pero mejorando el sistema de estimación y empleando la tecnología SVC, recientemente estandarizada como extensión de H.264/AVC y capaz de proporcionar contenidos adaptados a las condiciones que la red impone al usuario, mediante el escalado de las capas temporal y de calidad. Los parámetros observados de forma no intrusiva, como consecuencia de la transmisión del propio contenido multimedia, son empleados para realimentar el algoritmo diseñado al objeto de estimar los recursos disponibles para cada usuario. El algoritmo decide y actúa sobre el servidor de contenidos, adaptando la calidad del flujo a los recursos disponibles.

En este capítulo, el algoritmo de estimación se rediseña por completo para adaptarlo a la tecnología SVC. El sistema completo es evaluado empleando *streams* de vídeo reales

en escenarios emulados, lo que permite un excelente control de las condiciones de transmisión. Fue necesario desarrollar la aplicación servidor (con los módulos de estimación y adaptación para SVC), modificar la aplicación cliente para recoger las métricas necesarias y desplegar los escenarios para evaluar el sistema. El sistema se implementa en equipos reales y los resultados demuestran que la capacidad de adaptación del sistema es precisa, flexible y escalable, mejorando la QoE del usuario, así como a escalabilidad del sistema cuando al servicio acceden clientes de manera simultánea. Este hecho evidencia la viabilidad de un sistema de estas características.

11.1. Experimento 1: Evaluación de parámetros característicos del tráfico *streaming* en diferentes condiciones de ocupación de red

Al igual que el estimador basado en técnicas de transcodificación (capítulo 10), el principal problema en el planteamiento de sistemas adaptativos basados en codificadores escalables radica en la estimación del ancho de banda disponible. La solución al problema pasa por analizar de nuevo la red durante el proceso de transmisión del vídeo escalable, con el objeto de estudiar el comportamiento del tráfico ante diversas situaciones de congestión. A partir del análisis realizado podemos evaluar los parámetros de red que presentan una relación directa con el nivel de congestión y, por tanto, podrán ser empleados para realizar un algoritmo que permita calcular el ancho de banda disponible en cada momento, para el posterior ajuste de la calidad de los contenidos a las condiciones de transmisión.

En este primer experimento evaluaremos las características de la transmisión de contenido *streaming* generado con codificadores escalables, en este caso, SVC.

11.1.1. Definición de objetivos

Estudiaremos servicios de vídeo *streaming* con contenidos de alta calidad codificados en SVC, bajo una arquitectura simple de cliente-servidor en un entorno cableado, desde un punto de vista de investigación con el objetivo de evaluar el comportamiento de parámetros característicos de una sesión *streaming* cuando las condiciones de ocupación de la red varían.

11.1.2. Generación de preguntas y definición de métricas

Puesto que la tecnología subyacente para la transmisión del contenido multimedia sigue siendo RTP sobre UDP, evaluaremos las mismas métricas descritas en el apartado 10.1.2, esto es: pérdida de paquetes, jitter y linealidad. A partir de la experiencia adquirida en la realización de los experimentos del capítulo 10, no evaluaremos la métrica del GAP, puesto que no presenta diferencias sustanciales frente al *jitter*, siendo esta última una métrica más clásica y empleada ampliamente en la literatura.

11.1.3. Fase de preparación del escenario y de la recogida de datos

Para este experimento ha sido necesario el desarrollo un servidor *streaming* que soporte la transmisión de contenido H.264/SVC a través de RTP utilizando las librerías de *live555*. Al igual que en experimentos anteriores, emplearemos un equipo servidor, un equipo cliente y un tercer equipo emulando la línea de acceso. En este caso, el equipo encargado de emular la línea de acceso se configurará para emular diferentes capacidades de canal, empleando la herramienta NS-3[1]. Para obtener un rango de variación de los valores de 'q' (9.3) comprendido entre [0.6 - 1.6] es necesario combinar diferentes valores del ancho de banda del canal con la tasa de transmisión de los contenidos.

Por otro lado, se han empleado diferentes tipos de vídeo con diferentes contenidos, codificados en SVC mediante el software de referencia que proporciona la ITU (JSVM[2]) para, de esta manera, obtener un análisis más exhaustivo del tráfico. Se ha optado por incluir en el vídeo codificado dos tipos de escalabilidad: temporal y de calidad. La opción de la escalabilidad espacial no se contempla, ya que los cambios en el tamaño/resolución de la imagen durante el proceso de reproducción no proporcionan una buena experiencia de usuario. En el plano de escalabilidad temporal se dispondrá de 2 capas: la capa T0 y la capa T1, con valores de *frames* por segundo de 15 y 30 fps respectivamente. A su vez, a cada nivel temporal se puede añadir hasta 4 niveles extra de escalabilidad de calidad (desde Q0 a Q3) para cada nivel temporal, que permiten reducir el error de cuantificación de codificación y mejoran la PSNR. Para la escalabilidad de calidad hemos usado MGS (*Medium Grain Quality Scalability*), que nos permite descartar unidades de datos de las capas de mejora sin afectar al resultado del *bitstream* (Schwarz et. al, 2007). Realizamos los experimentos variando la capacidad de la línea para cada combinación de capas (TiQj).

El escenario planteado, así como los datos que son necesarios para el cálculo de las métricas, son análogos a los descritos en la sección 10.1.3 (Figura 11.1). La herramienta *tshark* nos permitirá capturar el tráfico generado tanto en el lado del cliente como en el lado del servidor. Además, una vez capturado el tráfico dispondremos de los datos necesario para el cálculo de las métricas, esto es: instantes de emisión y recepción del paquete, direcciones IP de origen y destino y los números de secuencia y valores de *timestamp* del paquete del protocolo RTP.

Por último, se comprueba y se valida todo el sistema antes de proceder con los experimentos.

11.1.4. Realización del experimento

Se procede a la realización de las pruebas, con varias repeticiones para cada combinación de la tasa de codificación de los contenidos y el factor de disponibilidad 'q'.

11.1.5. Recogida y validación de los datos

Una vez finalizada la realización del experimento procederemos al análisis de los datos recogidos con la herramienta *tshark*.

[1] http://www.nsnam.org/

[2] http://ip.hhi.de/imagecom_G1/savce/downloads/SVC-Reference-Software.htm.

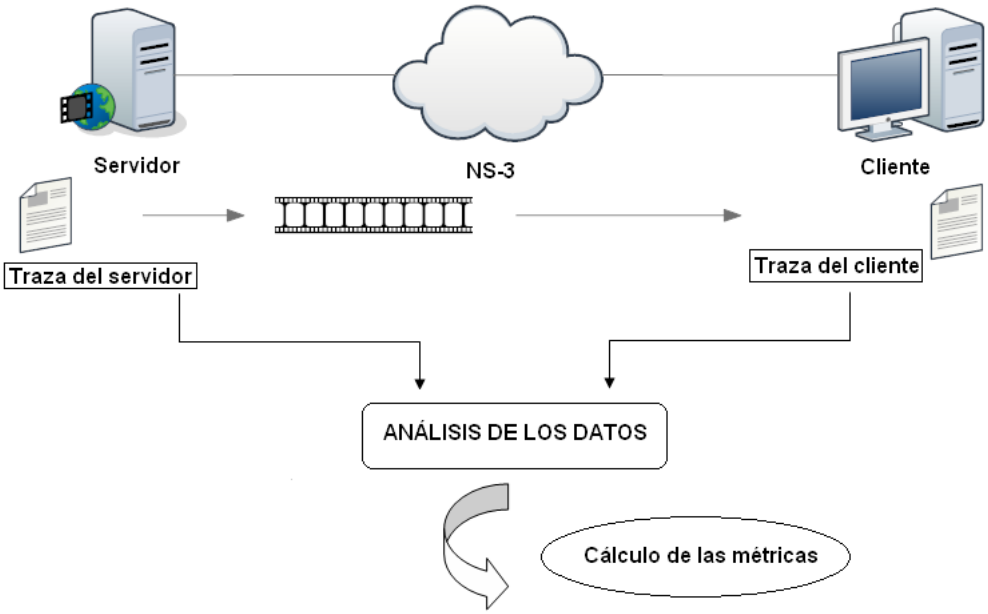


Figura 11.1: Experimento 1

11.1.6. Extracción de conclusiones

Los resultados de los experimentos de las variaciones de congestión para los codificadores escalables se resumen en los siguientes apartados. De nuevo, las métricas estudiadas presentan una dependencia con los niveles de ocupación en la red. A continuación analizamos en detalle los resultados obtenidos.

11.1.6.1. Pérdidas

Primeramente analizamos el porcentaje de pérdidas para diferentes valores de disponibilidad 'q' en el enlace y para diferentes combinaciones de escalabilidad temporal (T0-T1) y de calidad (Q0-Q3) de un *bitstream*. Cada una de las combinaciones supone diferentes tasas de codificación y por tanto diferentes consumos de ancho de banda.

La Figura 11.2 resume los resultados obtenidos. Al igual que los resultados mostrados en 10.1.6.1, la transmisión de contenidos codificados mediante técnicas escalables presenta también una clara dependencia entre los niveles de pérdidas de paquetes y el factor de disponibilidad 'q'.

11.1.6.2. Jitter

La Figura 11.3 muestra los resultados obtenidos para la métrica del *jitter* para cada combinación de capas, en función del nivel de congestión en la red. Se aprecia una mayor variación de la métrica del *jitter* frente a la ocupación de la red para valores de 'q' menores que la unidad. Concretamente, a medida que aumenta la congestión en la línea, el jitter sufre un incremento en su valor como consecuencia de los retardos producidos por la propia saturación del enlace.

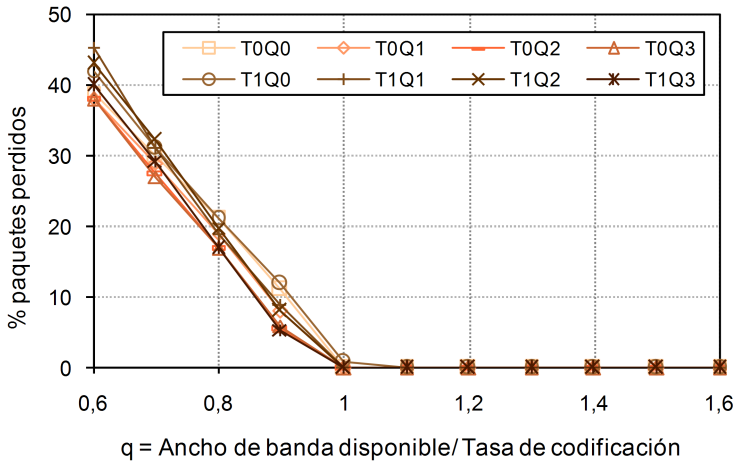


Figura 11.2: Porcentaje de pérdidas para distintos niveles de ocupación y distintas calidades de los contenidos

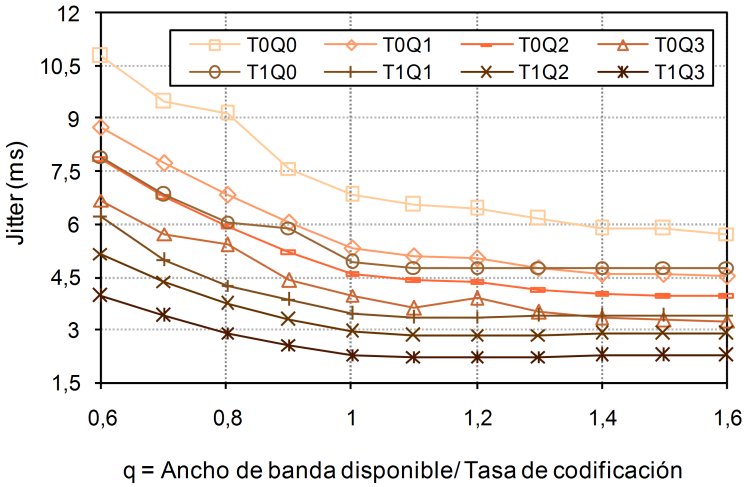


Figura 11.3: Jitter para distintos niveles de ocupación y distintas calidades de los contenidos.

11.1.6.3. Linealidad

Experimentos previos han demostrado que el análisis del coeficiente de correlación de Pearson para los instantes de llegada de los paquetes en recepción es un indicativo del nivel de ocupación en la red. A consecuencia de los retardos introducidos en una red altamente ocupada, la dispersión en los tiempos de llegada de los paquetes desdibujan el patrón a ráfagas recibido, propio de la transmisión *streaming* sin congestión mediante RTP. Los resultados del análisis para un sistema basado en codificadores escalables se muestran en la Figura 11.4.

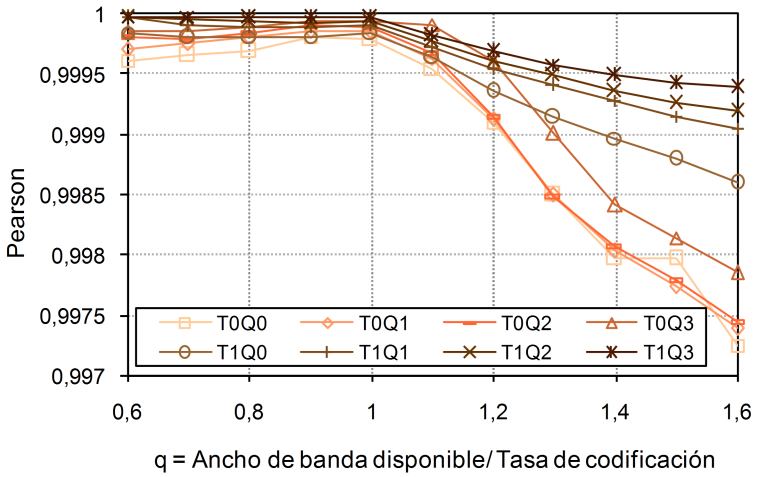


Figura 11.4: Linealidad en los tiempos de recepción para distintos niveles de ocupación y distintas calidades de los contenidos

Las curvas para cada una de las combinaciones de las capas temporales y de calidad presentan un comportamiento similar: valores próximos a la unidad para momentos en los que la red está congestionada, disminuyendo a medida que lo hace la ocupación en la red.

11.1.6.4. Resumen

Según el análisis realizado, el nivel de ocupación en la red influye en las métricas analizadas. El porcentaje de paquetes perdidos y el *jitter* se ven afectados por las situaciones de congestión principalmente (valores de 'q' menores que la unidad). Por el contrario, la métrica de la linealidad muestra variaciones claras para momentos en los que la red no presenta niveles de saturación.

A tenor de los resultados de las métricas analizadas para un sistema de transmisión *streaming* basado en codificadores escalables podemos concluir que, dichas métricas son factibles de ser empleadas como indicadores del nivel de ocupación en la red, con el fin de llevar a cabo posteriores procesos de adaptación de los contenidos transmitidos al ancho de banda disponible en la sesión *streaming*. En el siguiente apartado profundizaremos sobre esta idea.

11.2. Algoritmo de estimación

Los resultados obtenidos de la caracterización de la sesión *streaming* con codificadores escalables pueden ser empleados en el diseño de un algoritmo de estimación y adaptación para este tipo de escenarios y tecnología. A partir de la información obtenida de la caracterización de la transmisión de los contenidos, es posible determinar el ancho de banda disponible con la finalidad de adaptar las capas del *bitstream* de vídeo que se envían al cliente, maximizando la calidad percibida por el usuario y utilizando los recursos de red de manera eficiente. Cuando se detecte una situación de congestión deberemos disminuir

el *bit rate* de transmisión del vídeo, esto es, las capas del *bitstream* que se envían. Por el contrario, cuando no haya congestión, el *bit rate* podrá incrementarse.

A continuación explicaremos el diseño del algoritmo de estimación para un sistema basado en codificación SVC (*Scalable Video Coding*).

11.2.1. Diseño del algoritmo

Un servicio básico de *streaming* cuenta con un sistema cliente-servidor (y un proveedor de contenidos en aquellas arquitecturas que así lo requieran). En la arquitectura que planteamos es necesario tener en cuenta una serie de componentes adicionales a la arquitectura tradicional, tales como un módulo de estimación del ancho de banda y un módulo de adaptación de los contenidos para SVC (Figura 11.5), que permiten adecuar la tasa de transmisión de los contenidos (esto es, las capas del vídeo escalable que se envían al cliente) a los recursos libres de la red.

La solución propuesta en capítulos anteriores, y que mantendremos para el diseño del algoritmo del sistema basado en SVC, ha sido desarrollada sobre los protocolos propios para la transmisión de datos multimedia en una sesión de *streaming* (RTP/RTCP) y consiste en emplear los paquetes de *feedback* de RTCP para enviar información del cliente al servidor. Con esa información el servidor es capaz de identificar los estados de congestión en la red y actuar en consecuencia, respondiendo ante ellas por medio de una adaptación de las capas que se envían al cliente. Según este planteamiento, el módulo de estimación del servidor estima el ancho de banda disponible en la red a partir de la información que recibe del cliente para, a continuación, proceder a la adaptación de los contenidos mediante la determinación de la combinación óptima de capas a transmitir en función de dicha estimación, mediante el módulo de adaptación SVC.

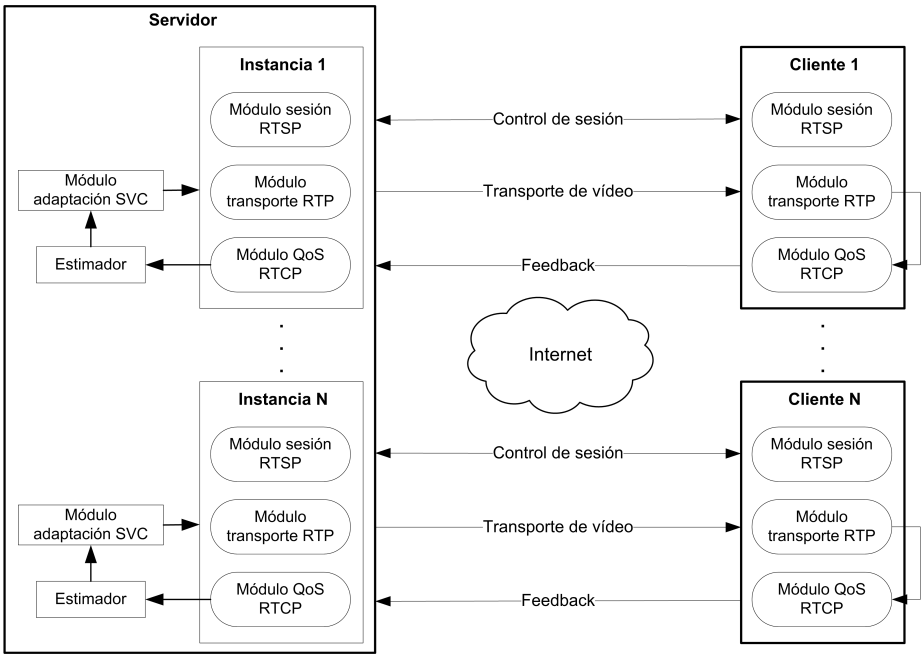


Figura 11.5: Arquitectura del sistema

La Figura 11.5 presenta la arquitectura del sistema propuesto. El cliente está compuesto por 3 módulos principales. Un módulo permite controlar la sesión multimedia («Módulo sesión RTSP»). Un segundo módulo se encarga de la recepción de los paquetes de vídeo («Módulo de transporte RTP»). El tercer módulo («Módulo QoS RTCP») analiza los paquetes de vídeo que llegan al cliente para calcular las métricas vistas en el apartado 11.1(paquetes perdidos, jitter y linealidad de los instantes de recepción) y encapsular el resultado en paquetes RTCP que se envían al servidor por medio de paquetes de control RTCP-APP. Para aumentar la estabilidad de los valores de las métricas, su cálculo se realizará utilizando la media móvil de los últimos 5 valores.

El uso de paquetes feedback RTPC-APP para estimar el ancho de banda disponible en un entorno no controlado como Internet presenta el problema de determinar la frecuencia de transmisión de dichos paquetes. Hemos realizado un estudio de la sobrecarga que se introduce en la red, variando la frecuencia de envío entre 2 y 10 segundos. En el peor de los casos, la sobrecarga introducida es menor del 0.1 % sobre el tráfico total. De ahí que podamos clasificar la técnica empleada para estimar el ancho de banda como no-intrusiva. Si el tiempo entre paquetes RTCP-APP es elevado, podría acarrear que la detección de situaciones de congestión se produjese demasiado tarde. Por otro lado, un tiempo entre paquetes APP de 2 segundos provoca situaciones de inestabilidad en el proceso de estimación. Los resultados indican que un periodo de 5 segundos es una buena solución de compromiso y proporciona resultados más estables, manteniendo el buffer del cálculo de métricas también en 5 segundos. Aún así, en los experimentos sucesivos de la implementación del algoritmo, evaluaremos diferentes configuraciones para el tiempo entre paquetes APP, puesto que, al liberar recursos en el servidor con respecto al sistema basado en transcodificación, podemos evaluar cambios más frecuentes en las tasas de codificación del contenido enviado.

Por otro lado, el servidor lo conforman además dos módulos encargados de llevar a cabo el proceso de estimación y adaptación de los contenidos respectivamente. Una vez que el paquete RTCP-APP llega al servidor, se procesa y, con la información que contiene, se determina el estado del enlace y las capas óptimas a transmitir (Figura 11.6).

La principal ventaja del empleo de la tecnología SVC en la arquitectura (además de reducir el volumen de almacenamiento en los servidores) se encuentra en la reducción del procesado computacional para llevar a cabo la adaptación de los contenidos en tiempo real. Esto es gracias al hecho de que SVC permite un manejo del *bitstream* de una manera sencilla gracias a la cabecera extra de 3 bytes que se añade con información adicional que contiene identificadores tales como D (Dependency ID), Q (Quality ID) y T (Temporal ID) que permiten llevar a cabo los procesos de adaptación (Amon et. al, 2007).

11.2.2. Estados

El análisis de las métricas recibidas acerca del estado de la transmisión nos permite clasificar los resultados en función de los niveles de ocupación de la red. Para un sistema basado en codificación escalable estableceremos dos estados diferentes: estado con congestión y estado sin congestión.

- Congestión: situación que se produce cuando la tasa de codificación es mayor que

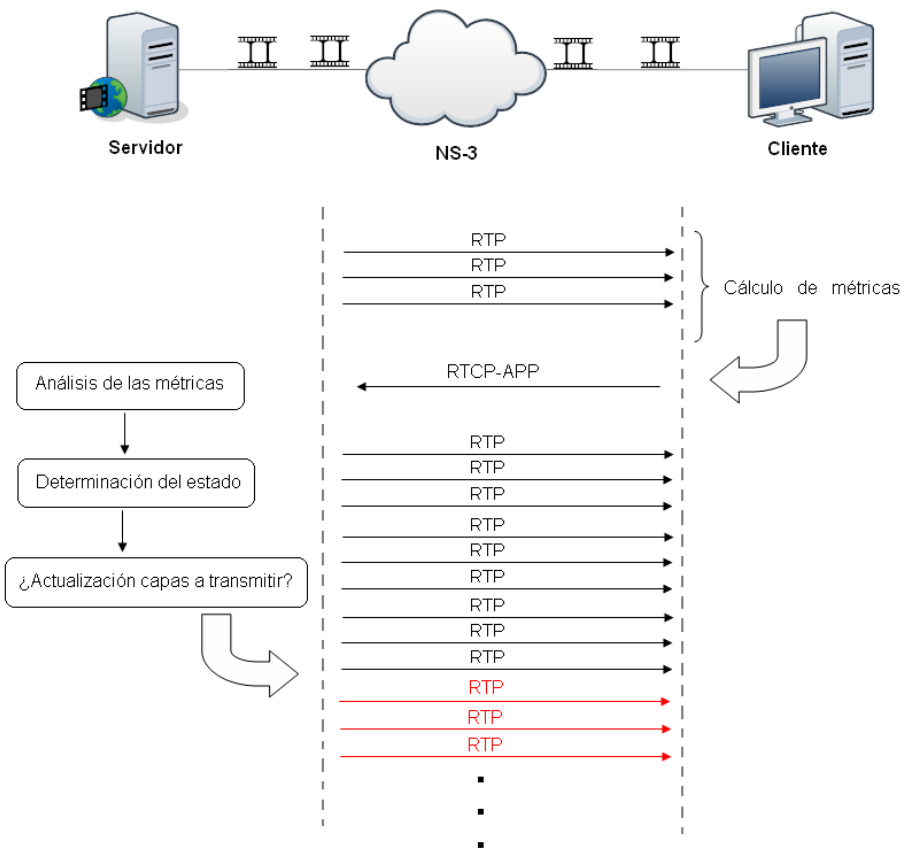


Figura 11.6: Diseño del sistema de adaptación

el ancho de banda disponible. Relacionándolo con la ecuación 9.3, se correspondería con valores de q<1.

- Sin congestión: cuando parte del ancho de banda de la conexión está libre. En este caso podemos utilizar el ancho de banda sobrante para mejorar la calidad del vídeo aumentando las capas de vídeo que se envían al cliente, siempre y cuando las capas que se añadan no superen la capacidad disponible en la red.

Esta clasificación marca una diferentecia con respecto a la clasificación de estados definida en la sección 10.2.2 para un sistema basado en transcodificación. Para un sistema basado en transcodificación, además, existía el estado de equilibrio, definido como una franja en la que no se llevaban a cabo cambios en la tasa de codificación, puesto que los contendidos transmitidos se adecuaban a los recursos libres en la red y no constituían la necesidad de realizar un incremento en el bitrate transmitido. El sistema de transcodificación presenta un sistema continuo, donde cualquier tasa de transmisión puede ser alcanzada y de ahí la necesidad de incluir el tercer estado de equilibrio para evitar ajustes de tasa innecesarios.

Esta situación cambia en SVC. El sistema de codificación escalable se traduce en un sistema discreto en cuanto a bitrates disponibles para la transmisión: cada combinación de capas presenta un bitrate definido y no existen valores intermedios. Por lo tanto, únicamente existen dos posibilidades a la hora de clasificar el estado de la transmisión:

que el ancho de banda disponible sea suficiente para el envío de las capas seleccionadas o que el ancho de banda sea insuficiente. Además, al ancho de banda estimado añadiremos una salvaguarda del 20 %.

11.2.3. Determinación del estado

Una vez que el cliente calcula las métricas (porcentaje de pérdidas, el *jitter* y el coeficiente de *Pearson* (o linealidad)), el servidor emplea dicha información para estimar el estado de la transmisión atendiendo a la clasificación anterior. A continuación se detalla dicho proceso.

11.2.3.1. Estimador basado en pérdidas, jitter y linealidad

Como hemos visto en la sección 11.1.6, las métricas descritas presentan una relación directa con el nivel de congestión. Por tanto, podemos emplear esa información para llevar a cabo procesos de adaptación de los contenidos multimedia al ancho de banda disponible, utilizando así los recursos de forma eficiente y mejorando la calidad percibida para las condiciones de transmisión dadas. Con la adaptación se pretende dotar de cierta QoS a los procesos de distribución de audio/vídeo a través de Internet, evitando las situaciones que producen pérdida de paquetes y que, como resultado, reducen la calidad causando un impacto negativo en la experiencia de usuario. Cuando se detecte una situación de congestión deberemos disminuir la tasa binaria de transmisión del vídeo, esto es, habrá que eliminar las capas que conforman los niveles más altos de calidad. Por el contrario, cuando no haya congestión, la tasa binaria podrá incrementarse, aumentando el número de capas que se envían al cliente.

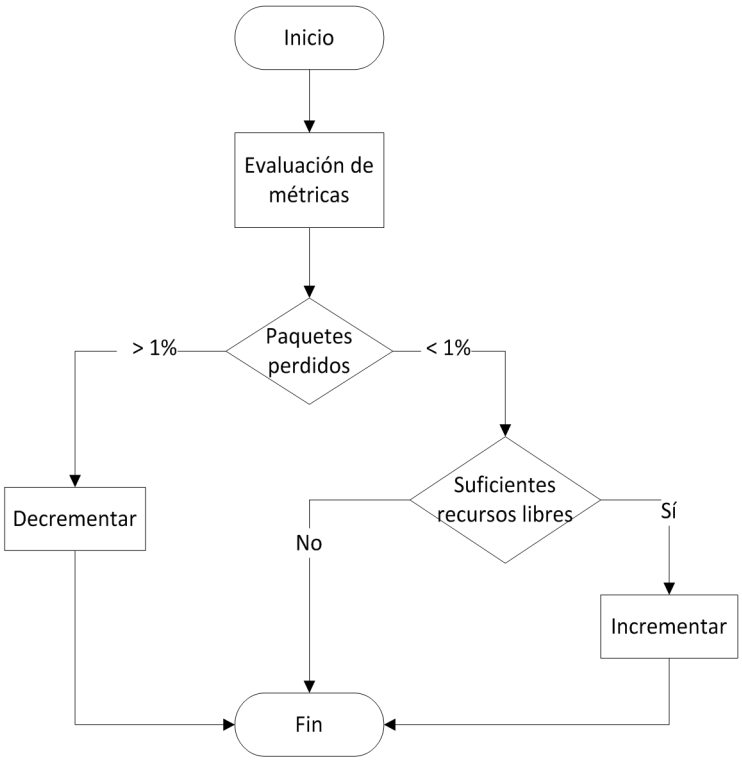


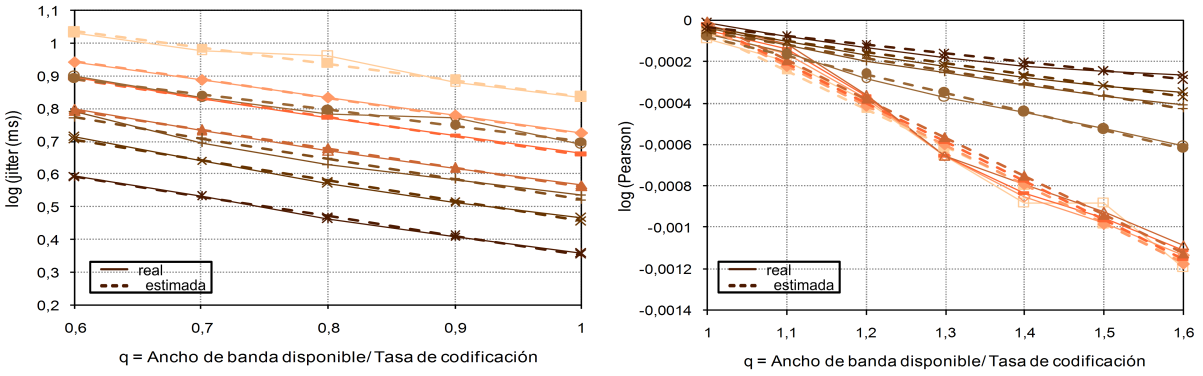
Figura 11.7: Diagrama de flujo del algoritmo de estimación

La Figura 11.7 describe el algoritmo empleado en el proceso de adaptación en el servidor, que incluye modificación en tiempo real de los niveles temporales y/o de calidad que se transmiten en función del estado del enlace. El servidor analiza el paquete RTCP-APP que recibe y que contiene el valor de las métricas (pérdidas, jitter, coeficiente de correlación de Pearson) calculadas por el cliente. El primer paso en la evaluación de las métricas será comprobar el porcentaje de pérdidas. Si éstas superan el umbral establecido del 1 %, se procederá a decrementar la tasa binaria de transmisión a base de eliminar capas en el vídeo transmitido. Si por el contrario las pérdidas son menores del 1 %, habrá que estudiar el estado de la transmisión para estimar si es posible aumentar la tasa de envío de los contenidos con capas de niveles superiores. Este análisis de los recursos libres es necesario, ya que la existencia de ancho de banda disponible en la red no implica que sea el suficiente como para añadir una capa de calidad a los contenidos transmitidos.

A partir de la ecuación 9.3 podemos estimar el ancho de banda disponible: la tasa de codificación es un parámetro conocido y 'q' se puede estimar a partir de las métricas vistas en la sección 11.1. De acuerdo a las Figuras 10.6 y 10.7, las curvas del *jitter* y las curvas del coeficiente de Pearson siguen una ecuación exponencial de la forma $A \cdot B^q$. En escala logarítmica hemos encontrado una tendencia lineal en estas curvas, tal y como muestra la Figura 11.8. Para caracterizar ambos gráficos mostrados en la Figura 11.8, hemos calculado a familia de parámetros $\left\{A^{Jitter}_{T_iQ_j},\ B^{Jitter}_{T_iQ_j}\right\}$, $\left\{A^{Pearson}_{T_iQ_j},\ B^{Pearson}_{T_iQ_j}\right\}$ para cada combinación de capas. De esta manera, la estimación de 'q' se realiza a partir de las ecuaciones 11.1 y 11.2

$$\begin{cases} log(Jitter) = log(A^{Jitter}_{T_iQ_j}) + q \cdot log(B^{Jitter}_{T_iQ_j},) \\ q = \frac{log(Jitter) - log(A^{Jitter}_{T_iQ_j})}{log(B^{Jitter}_{T_iQ_j})} \end{cases} \tag{11.1}$$

$$\begin{cases} log(Pearson) = log(A^{Pearson}_{T_iQ_j}) + q \cdot log(B^{Pearson}_{T_iQ_j},) \\ q = \frac{log(Pearson) - log(A^{Pearson}_{T_iQ_j})}{log(B^{Pearson}_{T_iQ_j})} \end{cases} \tag{11.2}$$



(a) Jitter (b) Linealidad

Figura 11.8: Curvas reales y estimadas utilizadas en el proceso de estimación

Cuando sea necesario aumentar la tasa de transmisión, se empleará la métrica del coeficiente de correlación de Pearson para estimar el ancho de banda disponible, ya que,

para este caso la métrica presenta variaciones cuando q>1 y por tanto, será una medida válida de estimación de los recursos libres en ese tipo de situaciones. Por otro lado, cuando sea necesario disminuir la tasa binaria (situación en la que las pérdidas presenten un valor superior al 1 %), el *jitter* será la métrica elegida para estimar el estado óptimo para la transmisión, puesto que para este caso, la región de interés en la que la métrica del *jitter* depende del nivel de ocupación de la línea será para valores de q<1.

11.3. Experimento 2: Evaluación de un estimador basado en pérdidas, jitter y linealidad

Una vez demostrada la necesidad de codificación adaptativa (9.2), y de realizar un estudio en profundidad del tráfico *streaming* para sistemas basados en codificación escalable SVC (11.1), pasamos a evaluar el sistema de estimación propuesto en 11.2.3.1.

11.3.1. Definición de objetivos

Estudiaremos un servicio de vídeo *streaming* adaptativo basado en SVC con contenidos de alta calidad, bajo una arquitectura simple de cliente-servidor con una línea de capacidad variable, desde un punto de vista investigador, con el fin de evaluar el comportamiento del estimador.

El estimador no deberá generar situaciones de oscilación, es decir, el ajuste de las capas de vídeo que se transmiten (bien sea aumentándolas o disminuyéndolas) tiene que hacerse únicamente cuando sea necesario y en el sentido correcto, evitando realizar, por ejemplo, ajustes de incremento de capas cuando en realidad tenemos una situación con congestión.

11.3.2. Generación de preguntas y definición de métricas

¿Cómo es el comportamiento del sistema adaptativo basado en SVC ante un canal cuya capacidad varía con el tiempo? ¿el algoritmo realiza estimaciones de manera correcta? ¿en situaciones de equilibrio se vuelven a generar ajustes?

Para contestar a estas preguntas deberemos plantear las métricas que será necesario evaluar durante los experimentos para dar respuesta a los interrogantes planteados. Al igual que se explicó en la sección 10.3.2, analizaremos el error cuadrátido medio (mse) entre la tasa de transmisión ideal y la tasa de transmisión estimada.

$$MSE(\hat{X}) = E\left[(\hat{X} - X)^2\right] \tag{11.3}$$

También deberemos comprobar la existencia de sobreestimaciones y su amplitud, puesto que este fenómeno tiene asociado la subsiguiente pérdida de paquetes en recepción.

11.3.3. Fase de preparación del escenario y de la recogida de datos

Cliente y servidor llevan a cabo una sesión *streaming* a través de una red emulada con NSE. La emulación permite un control absoluto sobre las condiciones del medio, permi-

tiendo variar en tiempo real las características del ancho de banda disponible. Para una completa evaluación del algoritmo de estimación y adaptación, se emplearán diferentes modelos de variación del canal.

Para modificar las condiciones de variación del ancho de banda del canal en un entorno controlado se inyectará tráfico de fondo CBR con el fin de congestionar el medio. El enlace de la red de acceso contará con una capacidad de 10Mbps y el tráfico CBR seguirá diferentes modelos de variación, descritos en el apéndice C.2. La variación del ancho de banda oscilará entre valores de 4Mbps y 10Mbps, con patrones de tráfico siguiendo una tendencia ascendente o descendente. De esta manera conseguimos caracterizar el comportamiento del estimador en diferentes situaciones.

El equipo servidor cuenta con los módulos de estimación y adaptación desarrollados para la tecnología SVC, y por su parte, el cliente incluye una versión modificada de openRTSP que incluye el módulo de cálculo de métricas a partir de la información de los paquetes RTP. Tanto el cliente como el servidor implementan el sistema de intercambio de información de control basado en paquetes RTCP-APP.

Para el análisis del comportamiento del estimador se empleará el vídeo "BRIDGE" obtenido de los repositorios de las secuencias de test de vídeo[3], y codificado con SVC siguiendo las indicaciones vistas en el apartado 11.1.3 y de acuerdo a objetivos de distribución de vídeo de alta calidad. La Figura 11.9 muestra el bit rate de cada capa, de acuerdo al nivel temporal (fps) y a las capas de calidad MGS (*Medium grain quality scalability*).

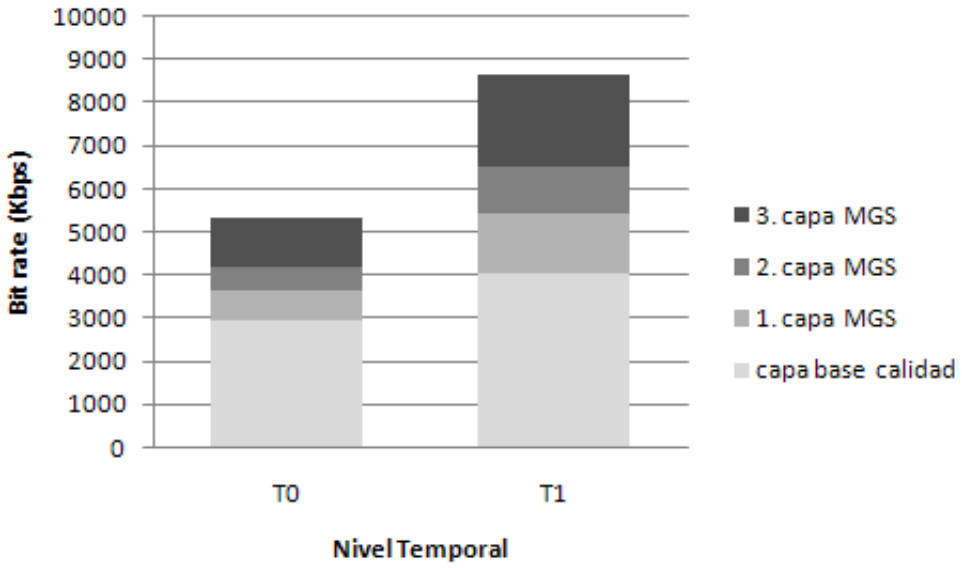


Figura 11.9: Características del vídeo empleado en las pruebas

El equipo hardware de la infraestructura propuesta está formado por dos servidores Dell 850 (1xdual core, 2.4GHz, 1GB) para el servidor de vídeo y la herramienta de emu-

[3]http://trace.eas.asu.edu

lación NS3, y un servidor Dell R410 (2xquad core, 2.13GHz, 16GB) para los clientes.

Con el fin de obtener resultados estables, se ejecutarán 50 realizaciones de cada experimento.

Los datos necesarios para calcular las métricas serán la capacidad de la red en cada instante y la tasa de codificación. Estos valores se recogerán en tiempo real en ficheros con información extraída directamente de las aplicaciones (Figura 11.10).

La duración máxima de cada experimento se establecerá en 60 segundos.

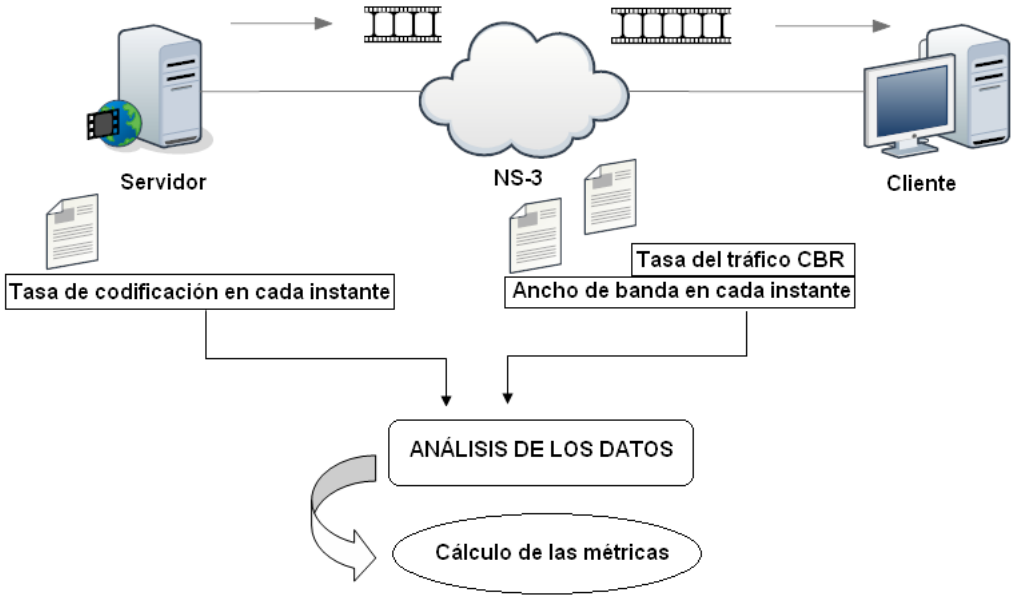


Figura 11.10: Experimento 2

Al igual que en experimentos anteriores, es necesario comprobar y validar todo el sistema antes de pasar a la siguiente fase en el desarrollo de la prueba.

11.3.4. Realización del experimento

Los detalles de la arquitectura y el esquema empleado para llevar a cabo todas las pruebas con los diferentes escenarios pertenecientes a este experimento pueden encontrarse en el apéndice D. Se realizaron 50 repeticiones para cada configuración del escenario, con el fin de obtener resultados estables.

11.3.5. Recogida y validación de los datos

Se analizan los datos capturados para proceder al cálculo de las métricas propuestas.

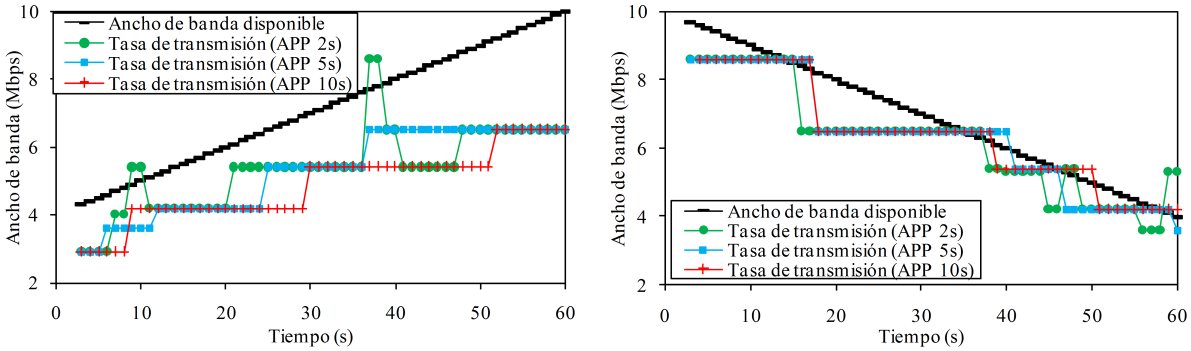
11.3.6. Extracción de conclusiones

A continuación se discuten los principales resultados obtenidos. Comenzaremos analizando algunos ejemplos gráficos del proceso de adaptación del algoritmo evaluado. Seguidamente, evaluamos las métricas propuestas para analizar el comportamiento de algoritmo: el error cuadrático medio y las sobreestimaciones.

11.3.6.1. Algunos ejemplos gráficos

A continuación mostramos algunos ejemplos gráficos del proceso de adaptación. Los gráficos se corresponden con situaciones estables, en las que hemos eliminado el transitorio inicial del proceso de estimación. Cada una de las gráficas presenta los resultados del proceso de estimación y adaptación para diferentes escenarios y con diferentes periodos de envío de paquetes APP (2 segundos, 5 y 10 segundos). El tamaño del *buffer* que emplea el cliente para calcuar las métricas será igual al tiempo entre los paquetes de *feedback*.

Para el primero de los escenarios proponemos un incremento/decremento progresivo del ancho de banda disponible. En la Figura 11.11a, donde los recursos libres presentan una tendencia incremental, las sobreestimaciones en el sistema son prácticamente nulas, excepto para el caso en el que el tiempo entre paquetes APP es de 2 segundos. Este caso presenta un comportamiento inestable en algunas situaciones. En general, las capas que se transmiten únicamente se incrementan cuando hay suficientes recursos en la red. Es necesario hacer notar que las tasas a las que es posible ajustar los contenidos tienen valores discretos (cada capa que forma el vídeo codificado en SVC tiene un determinado bitrate). Por esta razón, en la gráfica observamos que, en algunos intervalos de tiempo, pese a tener recursos libres de red, el algoritmo no realiza ajustes en las capas transmitidas, ya que añadir el siguiente nivel de calidad supondría exceder los límites del ancho de banda de red disponible, causando una situación de congestión.



(a) Rampa ascendente

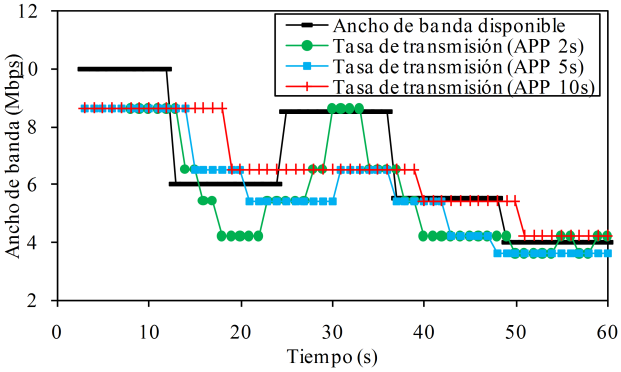
(b) Rampa descendente

Figura 11.11: Modelos rampa

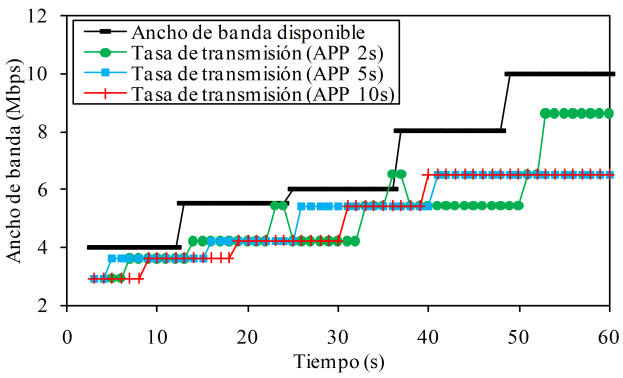
Si analizamos el caso de la Figura 11.11b podemos comprobar que cuando el estimador recibe el paquete APP con la información de la congestión en la red, automáticamente se eliminan las capas superiores que se están transmitiendo. La situación de congestión puede llegar a durar un máximo de 2, 5 o 10 segundos, en función del tiempo entre el

envío de paquetes RTCP-APP por parte del cliente.

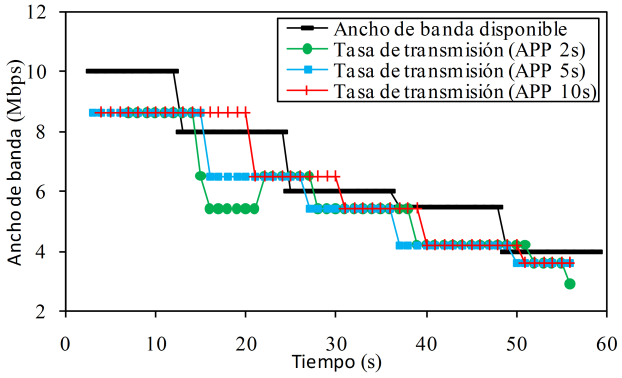
Otro aspecto importante que es necesario considerar a la hora de analizar el comportamiento del estimador se refiere a la estabilidad de las estimaciones, evitando situaciones de oscilación en la tasa de transmisión cuando la capacidad de la red presenta valores constantes. Para este estudio, proponemos los modelos de canal escalonados. En las Figuras 11.12a, 11.12b y 11.12c podemos observar que, una vez alcanzada una situación estable, el sistema no realiza nuevos ajustes de las capas que se transmiten hasta que no se produzca un cambio en la disponibilidad de los recursos. Sin embargo, cuando el tiempo entre paquetes APP es reducido (2 segundos), hay más oscilaciones en la tasa estimada.



(a) Escalón aleatorio



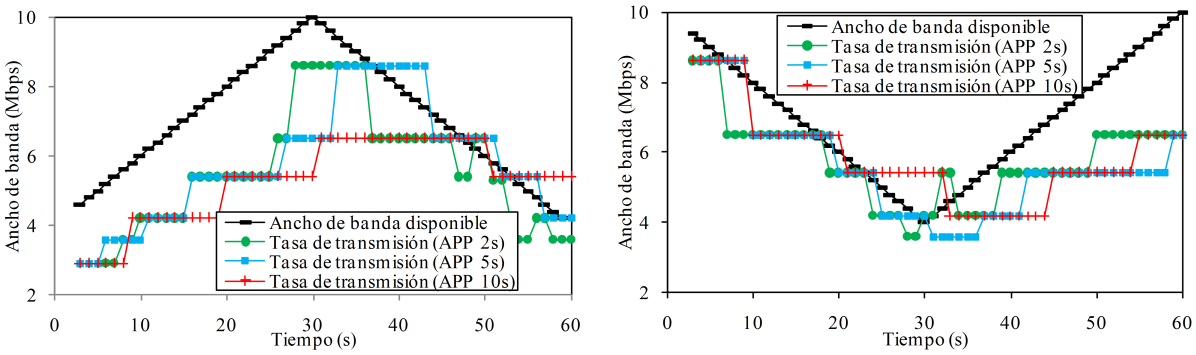
(b) Escalón ascendente



(c) Escalón descendente

Figura 11.12: Modelo escalón

Por último, también analizamos escenarios mixtos, donde se producen tanto incrementos como decrementos del ancho de banda disponible. En la Figura 11.13 se observa que el estimador realiza los ajustes correctamente y en la dirección adecuada: las capas que se transmiten se reducen cuando los recursos en la red disminuyen y, de la misma manera, cuando los recursos libres aumentan, las capas que se envían también se incrementan.



(a) Triangular ascendente (b) Triangular descendente

Figura 11.13: Modelo triangular

A la luz de los resultados gráficos podemos concluir que un periodo corto en el tiempo entre paquetes APP puede causar situaciones de inestabilidad en la estimación del ancho de banda disponible. Por otro lado, un tiempo de envío elevado entre dichos paquetes APP se traduce en una reacción lenta a los cambios en las condiciones de red. Por tanto, la solución final adoptada consiste en enviar los paquetes de feedback cada 5 segundos. Este periodo ha resultado ser el mejor para la mayoría de los escenarios propuestos. La respuesta ante cambios en las condiciones del enlace es un atributo importante en cualquier algoritmo de control de congestión y, como se observa en las figuras anteriores, nuestro sistema responde adecuadamente ante las variaciones en los recursos libres de la red, modificando los niveles temporales y de calidad del *bitstream* escalable que se transmite, con el fin de evitar la congestión.

Las siguientes tablas y figuras evalúan el comportamiento del algoritmo de estimación y congestión (con la configuración de 5 segundos entre paquetes APP) a lo largo de las 50 realizaciones de cada escenario propuesto.

11.3.6.2. Error cuadrático medio

La Tabla 11.1 se resumen los resultados de la media, la varianza y el intervalo de confianza al 95 % para el error cuadrático medio entre los valores de adaptación ideales y los que han sido estimados por el algoritmo. Las tasas de transmisión a las cuales se puede adaptar el contenido presentan valores discretos. Por esta razón, en este análisis hemos optado por calcular el error cuadrático medio entre dichas variables. Ésta es otra diferencia con el capítulo anterior (capítulo 10), donde los contenidos podían adaptarse a valores continuos y, por tanto, el error cuadrático medio del proceso de adaptación se calculaba entre el ancho de banda actual y la tasa de transmisión estimada.

En la propuesta que se realiza, la media del error cuadrático medio calculada para cada escenario presenta un valor máximo de 2.35 en el peor de los casos. El mejor y el peor caso dependerán fuertemente de las características de los escenarios. Por ejemplo, los resultados del error cuadrático medio muestran que los valores estimados de ancho de banda disponible están próximos a los ideales cuando el patrón del canal sigue una tendencia desdencente. Por otro lado, el peor caso lo presenta el canal con una variación con cambios tanto ascendentes como descendentes. Aún así, los resultados muestran una

mejora en el error cuadrático medio frente al sistema basado en transcodificación de los trabajos anteriores, donde el mse podía alcanzar valores en torno a 28.

	Media	Varianza	Intervalos de confianza
Modelo tendencia ascendente	1.46	0.05	[1.41 - 1.51]
Modelo tendencia descendente	0.24	0.01	[0.21 - 0.27]
Modelo escalonado	1.37	0.09	[1.31 - 1.44]
Modelo escalonado ascendente	1.75	0.21	[1.65 - 1.84]
Modelo escalonado descendente	0.56	0.01	[0.53 - 0.58]
Modelo triangular ascendente	2.35	0.18	[2.25 - 2.46]
Modelo triangular descendente	1.32	0.08	[1.25 - 1.39]

Tabla 11.1: Parámetros estadísticos del mse en el algoritmo de estimación

La Figura 11.14 muestra la observación menor, los cuartiles, la mediana y la observación menor y, en algunos casos, las observaciones que presentan valores fuera de rango. El objetivo de este tipo de gráficos es presentar la variabilidad que existe en torno a la media. Los resultados están dentro de una banda de valores razonable y están presentados en el mismo orden de magnitud que los resultados de la Tabla 11.1.

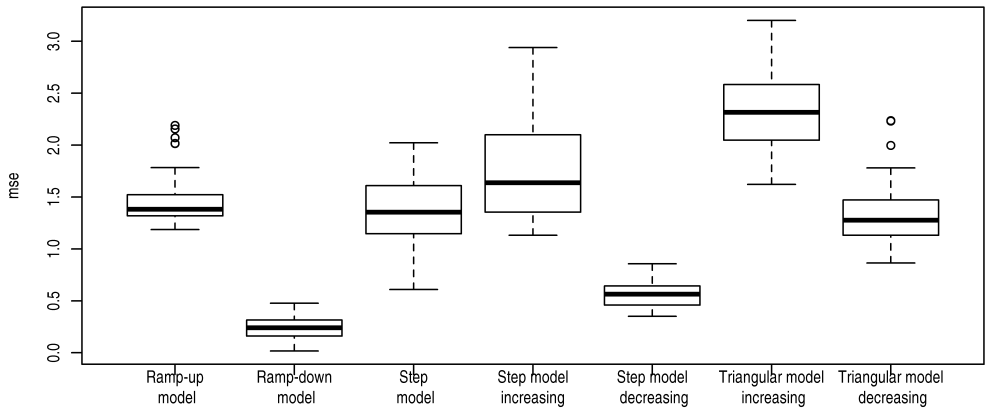


Figura 11.14: Boxplot para el error cuadrático medio del algoritmo de estimación

11.3.6.3. Sobreestimaciones

Las sobreestimaciones son otra medida importante en el análisis del estimador. La Tabla 11.2 y la Figura 11.15 muestran un resumen de la media, varianza, intervalo de

confianza al 95 % y boxplot del total de sobreestimaciones en cada escenario. El porcentaje de sobreestimaciones es prácticamente nulo en los modelos de canal que presentan un incremento continuado en el ancho de banda disponible.

	Media	Varianza	Intervalos de confianza
Modelo tendencia ascendente	0.23	0.85	[0.02 - 0.44]
Modelo tendencia descendente	8.93	17.08	[7.78 - 10.08]
Modelo escalonado	13.53	20.98	[12.52 - 14.54]
Modelo escalonado ascendente	0.15	0.97	[-0.06 - 0.35]
Modelo escalonado descendente	12.48	25.80	[11.46 - 13.50]
Modelo triangular ascendente	12.59	16.42	[11.60 - 13.58]
Modelo triangular descendente	12.41	25.33	[11.19 - 13.61]

Tabla 11.2: Parámetros estadísticos para las sobreestimaciones (%) en el proceso de adaptación

En el caso de modelos de canal que presentan algún tipo de variación descendente en cuanto a recuros libres, el porcentaje total de sobreestimaciones se incrementa. Esto es debido a que, después del proceso de adaptación, el estimador no recibirá un nuevo informe con las métricas provenientes del cliente hasta pasados 5 segundos. Durante este periodo, la capacidad disponible en la red puede verse reducida, dando lugar a situaciones en las que la tasa de envío de los contenidos supera al ancho de banda disponible.

Por último, también resumimos los resultados de la amplitud de las sobreestimaciones en la Tabla 11.3 y la Figura 11.16. El cálculo está basado en la diferencia entre el ancho de banda estimado y el disponible en las situaciones en las que existe congestión. Los valores están expresados en términos de porcentaje referido al ancho de banda total disponible. En general, la amplitud de las sobreestimaciones presenta valores bajos y de nuevo presentan relación con el modelo del canal: modelos de canal que tienen un incremento continuado en los recursos disponibles muestran una amplitud en las sobreestimaciones muy reducida, mientras que los modelos que tienen cambios más pronunciados también tienen mayor amplitud en las sobreestimaciones.

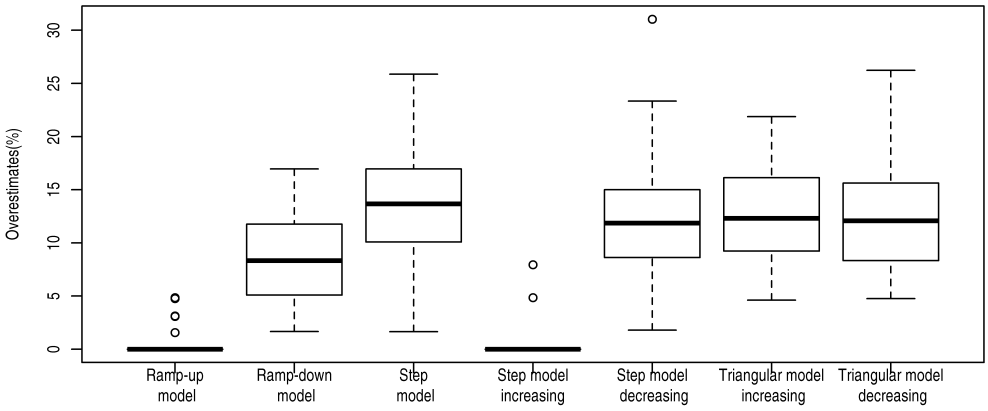


Figura 11.15: Boxplot para el porcentaje total de sobreestimaciones en el proceso de adaptación

	Media	Varianza	Intervalos de confianza
Modelo tendencia ascendente	0.18	0.47	[0.02 - 0.33]
Modelo tendencia descendente	5.43	14.59	[4.36 - 6.49]
Modelo escalonado	21.02	64.68	[19.24 - 22.80]
Modelo escalonado ascendente	0.11	0.56	[-0.05 - 0.27]
Modelo escalonado descendente	8.12	15.59	[7.33 - 8.92]
Modelo triangular ascendente	7.52	6.23	[6.92 - 8.13]
Modelo triangular descendente	5.71	2.42	[5.33 - 6.08]

Tabla 11.3: Parámetros estadísticos de la amplitud de las sobreestimaciones (%) en el proceso de adaptación

11.3.6.4. Tiempos de adaptación

Por último, evaluamos el tiempo necesario para llevar a cabo la adaptación y alcanzar una situación de estabilidad, donde no se realicen nuenvos ajustes en las capas transmitidas. Los resultados se muestran en la Figura 11.17. Para los casos en los que los recursos libres en la red se ven reducidos (valores negativos en el eje "variación en el ancho de banda"), el proceso de adaptación se realiza directamente sin la necesidad de realizar ajustes intermedios. En dichas situaciones, el tiempo total empleado siempre está por debajo de 6 segundos, incluyendo el tiempo para alcanzar la situación estable. Para los casos en los

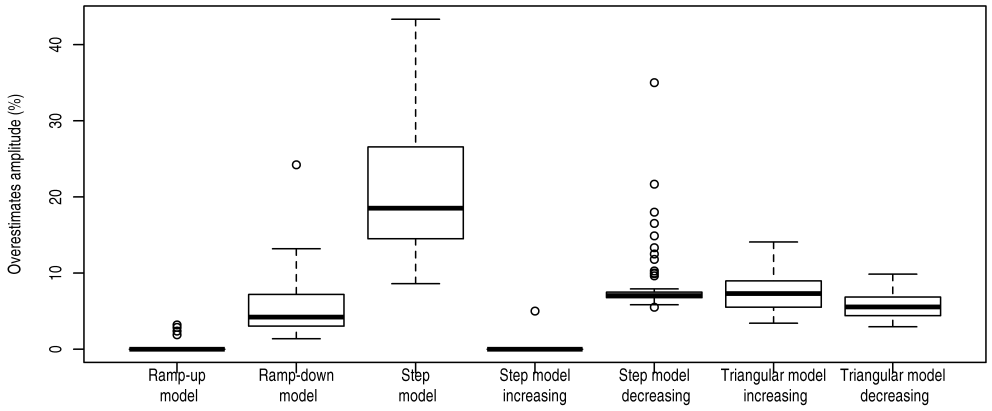


Figura 11.16: Boxplot para la amplitud de las sobreestimaciones (%) en el proceso de adaptación

que el ancho de banda disponible aumenta (valores positivos en el eje "variación en el ancho de banda"), la adaptación de los contenidos se realiza de manera progresiva, por lo que la situación de estabilidad se alcanza tras sucesivos ajustes en algunos casos.

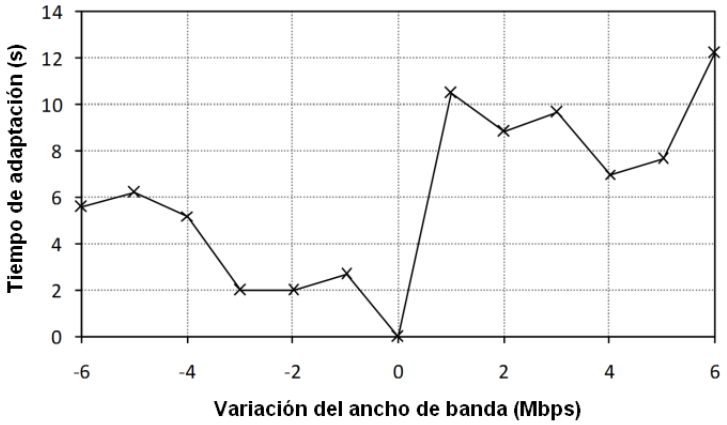


Figura 11.17: Tiempos de respuesta en el proceso de adaptación

11.4. Experimento 3: Análisis de escalabilidad del estimador

Otro factor clave y que marca una de las principales diferencias con el trabajo presentado en en capítulo previo (ver capítulo 10) será la escalabilidad del sistema. En el mencionado capítulo, debido a los procesos de transcodificación que tienen lugar para realizar el proceso de adaptación de los contenidos al ancho de banda estimado, el servidor se veía rápidamente sobrecargado, limitando el número de usuarios concurrentes que acceden al servicio.

En el siguiente experimento analizaremos la escalabilidad del sistema de estimación y adaptación basado en SVC.

11.4.1. Definición de objetivos

Estudiaremos servicios de vídeo *streaming* adaptativos con contenidos de alta calidad, bajo una arquitectura simple de cliente-servidor en un entorno cableado, desde un punto de vista de investigación con el propósito de evaluar la escalabilidad del sistema.

11.4.2. Generación de preguntas y definición de métricas

¿Cuántos usuarios es capaz de soportar el servicio?

La métrica para evaluar la escalabilidad será el número de usuarios concurrentes.

11.4.3. Fase de preparación del escenario y de la recogida de datos

En el estimador basado en codificación SVC, los procesos de transcodificación para adaptar la tasa de transmisión a los recursos disponibles no son necesarios, quedando las tareas de adaptación reducidas a la decisión del envío de los paquetes en función de la capa a la que pertenezcan. En este sentido, la limitación ahora estará en el tráfico real que es capaz de manejar correctamente el emulador de red NS3. Los autores (Alvarez et. al, 2010) realizan dicho estudio, llegando a la conclusión de que, para un escenario sencillo, el tráfico que puede llegar a gestionar NS-3 está en torno a 20 Mbps.

Por todo lo dicho, se recurrirá a una segunda opción de emulación de ancho de banda disponible basada en el conformado de tráfico, empleando TBF (Token Bucket Filter). Para este caso, el ancho de banda del canal será de 100 Mbps. Si se tiene en cuenta que la combinación de capas consideradas en el servidor y que producen la menor tasa binaria en el vídeo empleado en las pruebas es de 2.9 Mbps se obtiene que, teóricamente, el número máximo de clientes concurrentes accediendo al servicio para ese ancho de banda disponible total es de 34 clientes.

Para un correcto funcionamiento del sistema, el porcentaje de uso de CPU tendrá que estar por debajo de los valores críticos. De la misma forma, en el cliente también analizaremos y comprobaremos el porcentaje de uso de CPU. Para ello emplearemos la herramienta SAR.

En este punto resultará interesante destacar las características físicas de los equipos empleados. El equipo hardware de la infraestructura propuesta está formado por dos servidores Dell 850 (1xdual core, 2.4GHz, 1GB) para el servidor de vídeo y la herramienta de emulación NS3, y un servidor Dell R410 (2xquad core, 2.13GHz, 16GB) para los clientes.

Los clientes simultáneos se han creado con ayuda de la tecnología LXC[4].

[4]http://lxc.sourceforge.net

11.4.4. Realización del experimento

Para el escenario de evaluación descrito, se procede a la ejecución de los experimentos. Los clientes acceden de manera escalonada al servicio.

11.4.5. Recogida y validación de los datos

Todos los datos extraídos durante el proceso de ejecución de los experimentos son analizados para el cálculo de los parámetros de evaluación.

11.4.6. Extracción de conclusiones

En la Figura 11.18 mostramos la evolución del porcentaje de uso de CPU en el servidor por parte del proceso del servidor de *streaming* SVC adaptativo en función del número de clientes concurrentes accediendo al servicio. Para mantener el uso de CPU por debajo de los límites aceptados de 80 %, el número máximo de clientes concurrentes en el servidor es de 18. Si comparamos estos resultados de escalabilidad con el sistema previo basado en transcodificación (capítulo 10) observamos una mejora notable en el sistema, puesto que para la solución anterior el porcentaje de uso del 66.282 % se alcanzaba con únicamente 3 clientes simultáneos en el servicio.

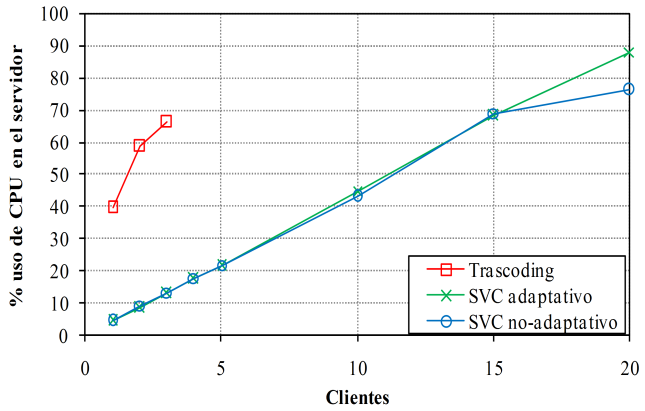


Figura 11.18: Comparativa del uso de CPU en diferentes sistemas

Cuando al sistema acceden múltiples usuarios de manera concurrente, para cada uno de ellos se crea un proceso de estimación de los recursos libres teniendo en cuenta el resto del tráfico presente en la red, tanto si es tráfico de otros usuarios de video *streaming* como si es tráfico de otras aplicaciones. Además, los procesos de subida de la tasa de transmisión siguen una postura conservadora, de manera que se pretende que exista una salvaguarda, evitando consumir el 100 % de los recursos libres de red. Por otro lado, en el proceso de evaluación de la escalabilidad que se describe, los clientes acceden al servicio de manera escalonada.

Podemos realizar el experimento en las mismas condiciones, pero con un sistema no adaptativo, donde se envíe siempre la misma combinación de capas. En concreto se van a estudiar un caso en el cual siempre se envíe la combinación de capas T1Q3 (30 fps con

el máximo nivel de calidad, 8.6 Mbps). Como observamos en la Figura 11.18, no existen grandes diferencias entre el sistema no adaptativo y el sistema adaptativo en cuanto al consumo de recursos en el servidor. La limitación en este caso viene impuesta por las elevadas tasas de transmisión que se manejan y, por consiguiente, en el número de usuarios concurrentes accediendo al servicio, y no a causa del ancho de banda disponible.

Por tanto, podemos concluir que nuestro sistema muestra el mismo nivel de escalabilidad que un sistema que no implementa la tecnología de estimación y adaptación. Es por eso que podemos afirmar que la solución propuesta y los algoritmos implementados no deterioran el funcionamiento del sistema, a diferencia del sistema previo basado en transcodificación.

11.5. Experimento 4: Evaluación de la QoE en SVC

La transmisión de servicios de vídeo sobre redes IP es, al igual que la VoIP, altamente sensible a degradaciones de la calidad. Esta pérdida de calidad puede ocurrir durante el proceso de codificación, en el proceso de transmisión de los paquetes a través de la red y/o durante el proceso de decodificación y reproducción.

En los sucesivos apartados se plantea un experimento que nos permite evaluar la calidad de experiencia (QoE) para un sistema adaptativo con las características propuestas.

11.5.1. Definición de objetivos

Estudiaremos la calidad de la experiencia de un sistema basado en codificación escalable SVC.

11.5.2. Generación de preguntas y definición de métricas

La evaluación de la calidad de los contenidos puede hacerse a través de métodos subjetivos (encuestas) o métodos objetivos (empleando algoritmos que calculen el nivel de calidad).

La desventaja de los métodos subjetivos se encuentra en la necesidad de un grupo de participantes que visionen y evalúen diferentes contenidos, así como el tiempo necesario para llevar a cabo este tipo de pruebas. En estos test, la calidad del vídeo se puntúa en una escala del 1 (inacpetable) al 5 (excelente), conocidos como escala MOS, tal y como ya hemos introducido en la sección 9.2.2.

Para las evaluaciones objetivas se emplean algoritmos. El más utilizado para evaluar la calidad del vídeo es la métrica de la PSNR, ya explicada (sección 9.2.2). Debido a su amplia aceptación en la comunidad científica, hemos optado por analizar el valor de la PSNR como métrica de evaluación de la calidad del vídeo recibido.

11.5.3. Fase de preparación del escenario y de la recogida de datos

El análisis de la PSNR, para el sistema adaptativo basado en SVC, requiere el desarrollo de una herramienta que permita manejar correctamente la ausencia de *frames* en el receptor debida a la eliminación de capas como consecuencia de la congestión en la red. Por ejemplo, si únicamente el servidor está enviando el nivel temporal 0 (T0), el cálculo de la PSNR tiene que tener en cuenta esta falta de *frames*. En caso contrario, el cálculo no será correcto, puesto que estaría comparando *frames* entre la fuente y el destino que no se corresponden. En la Figura 11.19a podemos ver un ejemplo en el que los *frames* enviados y los recibidos no están sincronizados para el cálculo de la PSNR y, por tanto, el resultado será erróneo. El escenario correcto sería el mostrado en la Figura 11.19b.

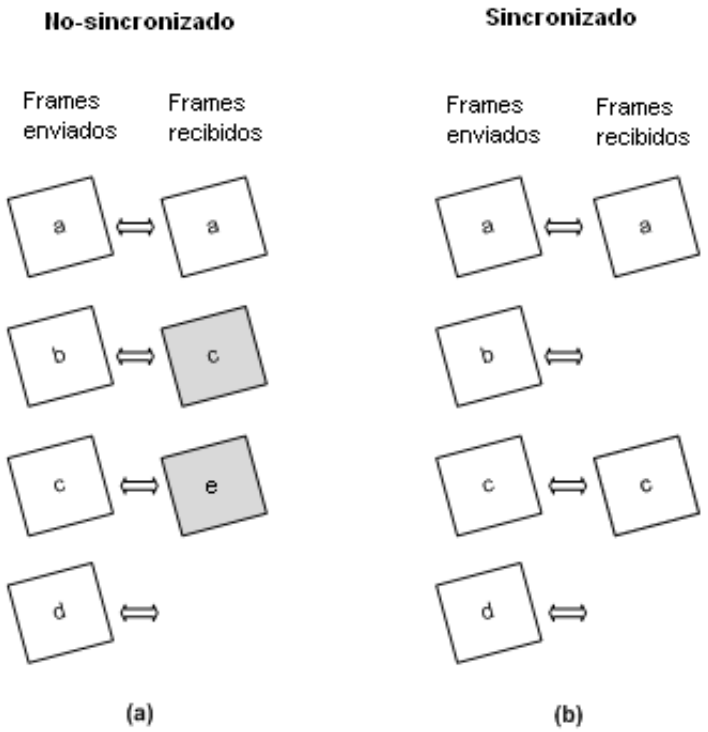


Figura 11.19: Ejemplo de sincronización de *frames*

11.5.4. Realización del experimento

Se ejecutan los experimentos en el escenario propuesto.

11.5.5. Recogida y validación de los datos

Se analizan los datos capturados para calcular las métricas de evaluación definidas anteriormente.

11.5.6. Extracción de conclusiones

Podemos analizar los valores de PSNR para vídeos con combinaciones fijas de las capas transmitidas, esto es, podemos comparar el contenido original con vídeos que presenten diferentes niveles de escalabilidad. De esta manera, podemos calcular la degracación de la calidad debida a los procesos de codificación y debido a los procesos de eliminar las capas de mejora.

Continuando con el ejemplo presentado en las secciones previas, el vídeo codificado estará compuesto de 2 niveles temporales con 4 niveles extra de calidad. La Figura 11.20 muestra el incremento de los valores de PSNR en función del número de capas de calidad que conformen los contenidos (desde Q0 a Q3). La escalabilidad de calidad reduce el error de cuantificación y mejora la PSNR. Los análisis de los resultados muestran la cualidad *no-reference* de los *frames* tipo B. Gracias a ella, la ausencia de paquetes únicamente afecta a la resolución temporal y no afecta a la estructura básica del vídeo. Como resultado, a pesar de la reducción en la resolución temporal, los valores de PSNR que se obtienen son elevados (por encima de 38 dB). Esta característica proporciona una resistencia a errores de la que otros códecs carecen. Para el análisis de la PSNR del vídeo con el nivel temporal más bajo, hemos reproducido el comportamiento del reproductor, donde la ausencia de *frames* pertenecientes a los niveles temporales superiores se solventa copiando el *frame* anterior.

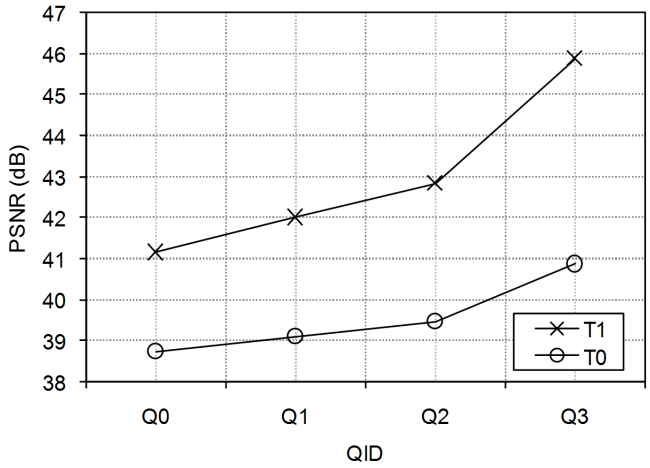


Figura 11.20: Degradación de la calidad debido al proceso de adaptación

11.6. Conclusiones finales

Con el fin de solucionar las limitaciones que presenta el estimador basado en técncias de transcodificación propuesto en el capítulo anterior, planteamos ahora un nuevo sistema de estimación y adaptación basado en codificadores escalables.

La experiencia desarrollada en el diseño e implementación de este tipo de sistemas nos conduce a desarrollar un nuevo análisis del tráfico *streaming* entre cliente y servidor, esta

vez empleando contenidos codificados en SVC. De nuevo obtenemos una clara dependencia de los valores de paquetes perdidos, *jitter* y linealidad de los instantes de recepción en función del nivel de congestión en la red.

Las ideas planteadas en el capítulo anterior para el diseño del sistema adaptativo se adecúan ahora al sistema basado en SVC, con resultados notablemente mejorados en lo que respecta a las cuestiones de escalabilidad en el lado del servidor: mediante el uso de codificadores escalables, conseguimos reducir la carga de procesamiento en el servidor para realizar las tareas de adaptación al ancho de banda estimado.

Capítulo 12

ESTIMACIÓN Y ADAPTACIÓN PARA HTTP STREAMING

En los capítulos previos hemos enfocado el problema de la calidad de experiencia del usuario desde el punto de vista de adaptación de los contenidos al ancho de banda, estudiando y proponiendo algoritmos de estimación y adaptación para dos soluciones diferentes: primeramente, transcodificación del contenido y, a continuación, el uso de codificadores escalables como SVC que permitían hacer el sistema más escalable que el primero. Los resultados del sistema basado en SVC muestran que la solución evaluada se adapta de manera rápida y precisa a los cambios en el ancho de banda disponible, evitando situaciones indeseadas de oscilación en las tasas de transmisión ajustadas. Sin embargo, esta solución presenta dos problemas principalmente: uno de ellos está relacionado con la sobrecarga en el servidor, puesto que la lógica de adaptación reside en él (a pesar de mejorar notablemente los resultados frente al primer sistema del capítulo 10). El otro problema se refiere a la necesidad de una infraestructura específica para la distribución de los contenidos, lo que lleva asociado altos costes de despliegue.

Cambiando el modelo del streaming de audio y vídeo sobre RTP, recientemente surge una nueva generación de aplicaciones *streaming* basadas en HTTP empleando la tecnología DASH *(Dynamic Adaptive Streaming over HTTP*). DASH permite la transmisión de vídeo *streaming* adaptativo sobre Internet usando el protocolo HTTP. Su gran ventaja radica en la facilidad a la hora de realizar el despliegue del servicio, proporcionando además servicios *streaming* para usuarios con condiciones de red dinámicas y dispositivos heterogéneos. En DASH, el contenido se codifica en múltiples versiones con diferente *bitrate*. Cada una de las versiones se divide a su vez en pequeños fragmentos de vídeo, que se envían al cliente por medio de servidores HTTP estándar. A diferencia de los sistemas anteriores (capítulos 10 y 11), la lógica de adaptación reside en el lado del cliente, y se simplifica en gran medida el algoritmo de estimación y adaptación. Los detalles acerca de la tecnología DASH se resumen en el capítulo 6.4.

El *streaming* sobre HTTP está siendo adoptado gradualmente por los proveedores de contenidos y de servicios en la red gracias a las ventajas en términos de calidad percibida y utilización de los recursos. A esto hay que añadir que el uso de conexiones HTTP/TCP permite reutilizar la infraestructura de red existente, aprovechando también la existencia de los proxies y las cachés. Además, se evitan los problemas típicos de RTP y UDP rela-

cionados con los *firewalls* y el protocolo NAT. El *streaming* de vídeo empleando HTTP se ha convertido en un fuerte candidato para la transmisión de vídeo en Internet. Y con la reciente estandarización del nuevo *MPEG Dynamic Adaptvie Streaming over HTTP* (DASH), la flexibilidad y la implantación de sistemas de vídeo adaptativos se ha visto incrementada debido al hecho de que DASH puede operar en una infraestructura web convencional.

En este capítulo se propone un cambio de perspectiva con respecto a los capítulos anteriores, diseñando e implementado un módulo de estimación y adaptación para la tecnología DASH. La forma en la que el cliente realiza la estimación del ancho de banda y la forma en la que, posteriormente, toma las decisiones de adaptación de los contenidos no están especificados en el estándar DASH, por lo que en este capítulo cubriremos ese vacío en la tecnología. En nuestra propuesta, el cliente solicitará los contenidos al servidor basándose en los niveles de ocupación del *buffer* y en la estimación del ancho de banda disponible que se realiza a partir de medidas del *throughput* (o caudal). Basándose en los valores de estas métricas, un algoritmo será el responsable de calcular el segmento adecuado que es necesario solicitar al servidor en función de los valores de las métricas. A diferencia de los sistemas anteriores, la lógica de adaptación se implementa íntegramente en el lado del cliente, de manera que se consigue reducir la carga en el servidor. Además, el método propuesto de adaptación no requiere información adicional de *feedback* entre el cliente y el servidor. Por último, puesto que la transmisión se basa en el protocolo TCP, emplearemos métricas clásicas de estimación del ancho de banda en el cliente, no siendo necesario llevar a cabo un proceso previo de análisis del tráfico generado entre el cliente y el servidor.

12.1. Algoritmo de estimación

A continuación se describe el diseño e implementación de un sistema de estimación y adaptación para DASH. El algoritmo adaptativo propuesto está basado en umbrales en el *buffer* del cliente y una medida suavizada del *throughput* (esto es, una medida basada en el *throughput* de segmentos previos).

Nuestro servicio de HTTP *streaming* sigue la arquitectura propuesta por el estándar DASH, con la lógica de adaptación implementada en el lado del cliente. Puesto el estándar DASH no define la implentación del diseño del sistema de adaptación, nosotros proponemos dos módulos diferenciados para llevar a cabo el proceso de, por un lado, estimación del ancho de banda disponible y, por otro lado, el proceso de selección del segmento acorde a la disponibilidad de recursos.

En las siguientes secciones procedemos a describir en detalle los módulos diseñados.

12.1.1. Diseño del algoritmo

Tal y como hemos analizado en los capítulos acerca de las consideraciones teóricas de las tecnologías empleadas (ver sección 6.4), en el estándar DASH nos encontramos con dos elementos diferenciados en lo que se refiere al contenido multimedia almacenado en el

servidor HTTP. Por un lado se encuentra el fichero MPD *(Media Presentation Description)* y otro elemento lo constituyen los segmentos de contenidos.

El fichero MPD es un fichero XML que describe las características de las diferentes versiones de los contenidos que están disponibles para ser descargadas en el servidor. DASH contempla la existencia de varias versiones (denominadas *representaciones*) disponibles en el servidor. Cada una de las representaciones es una opción en cuanto a características de codificación se refiere. Pueden existir diferentes representaciones de los contenidos que difieran en la tasa de codificación o en la resolución espacial o temporal.

El contenido multimedia, por su parte, se divide en segmentos. Cada segmento dispone de su propia URL y es auto-contenido, por lo que el reproductor del cliente puede solicitar y reproducir cada segmento de manera individual.

Sin embargo, repetimos que el estándar DASH no contempla el diseño de los mecanismos de adaptación que el cliente puede emplear para decidir qué segmento solicitar. La especificación MPEG-DASH únicamente hace referencia al formato del fichero MPD y el formato de los segmentos. El envío del propio fichero MPD al cliente, la codificación de los contenidos o el comportamiento del cliente están fuera de la especificación del estándar. Por todo lo dicho, la arquitectura que se propone incluye un sistema de adaptación para seleccionar el segmento más apropiado para la descarga y posterior reproducción, atendiendo al ancho de banda estimado.

La Figura 12.1 resume un ejemplo del escenario y la arquitectura entre un servidor HTTP y un cliente DASH, destacando los módulos que son parte del diseño propuesto. Como se observa, el sistema de adaptación mencionado consta de dos sub-sistemas que se corresponden con los algoritmos de estimación del ancho de banda disponible y el algoritmo de selección de segmentos.

En la solución propuesta, el cliente solicita primeramente el archivo MPD, con el fin de disponer de toda la información relativa a las características de los contenidos multimedia del servidor y las diferentes representaciones. En los instantes iniciales, el cliente solicitará los segmentos que se correspondan con la representación cuyo bitrate sea el menor disponible. La solicitud de los segmentos se realiza mediante peticiones HTTP. A partir del flujo de paquetes TCP, el cliente realiza las estimaciones del ancho de banda disponible. A continuación, los siguientes segmentos seleccionados para la descarga se eligen en función del resultado del ancho de banda estimado por el módulo de estimación y en función del nivel de ocupación del *buffer* del cliente.

12.1.1.1. Métricas empleadas

El sistema adaptativo propuesto basado en DASH emplea TCP como protocolo de transporte. Los sistemas previos presentados en los capítulos 10 y 11 empleaban un sistema clásico de *streaming* basado en RTP sobre UDP. Esta gran diferencia marca el cambio en las métricas empleadas.

El sistema de adaptación diseñado está dividido en dos módulos claramente diferenciados, cada uno de ellos con un cometido concreto en el proceso de adaptación. Por tanto,

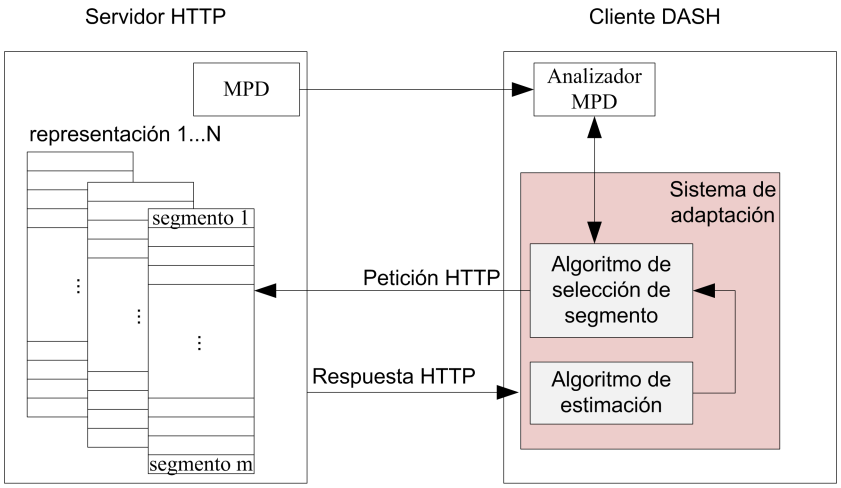


Figura 12.1: Arquitectura del sistema

cada módulo empleará diferentes criterios o métricas.

1. Algoritmo de estimación de ancho de banda disponible: la métrica empleada en el algoritmo de estimación estará basada en el la media del *throughput* de los segmentos previos recibidos.

   La estimación del *throughput* se realiza mediante el cociente de los *bytes* transmitidos y el tiempo necesario para transmitir esos *bytes* al destino. El cálculo se realiza en el cliente DASH, extrayendo la información del tamaño del segmento a partir del campo «*Content-Length*» de la cabecera HTTP. Al igual que hemos hecho en los estimadores previos, suavizaremos el resultado obtenido con la media de los últimos 5 valores del *throughput* estimados .

2. Algoritmo de selección de segmento: el módulo de selección de segmento basará sus decisiones en base a dos criterios: el ancho de banda estimado (obtenido en el módulo que implementa el algortimo de estimación, explicado anteriormente) y el nivel de ocupación del *buffer* del cliente.

   El empleo de la métrica relacionada con el ancho de banda disponible estimado resulta obvio para poder decidir qué versión de los contenidos se descargan. Completamos la decisión con el análisis del estado del *buffer* en el cliente.

En los sucesivos apartados profundizaremos acerca del funcionamiento del sistema de estimación y adaptación en función del valor de las métricas descritas.

12.1.2. Estados

A partir de los valores de las métricas, podemos definir dos estados diferenciados, directamente relacionados con el nivel de ocupación en la red:

- Congestión: situación a evitar y que debe ser corregida en la mayor brevedad posible.
- Sin congestión: estado óptimo al que el algoritmo diseñado debe converger.

La definición de los estados propuesta para el sistema adaptativo basado en DASH es similar a la clasificación detallada en la sección 11.2.2 para un estimador basado en codificadores escalables, por lo que no extenderemos más los detalles de la descripción.

12.1.3. Determinación del estado

El análisis de las métricas seleccionadas nos permitirá determinar el estado de la transmisión y la correspondiente acción que deberemos llevar a cabo.

La Figura 12.2 describe el diagrama de flujo de los algoritmos empleados en el proceso de adaptación en el lado del cliente. Primero, el algoritmo de estimación calcula el ancho de banda disponible. A continuación, el algoritmo de selección de segmento selecciona el segmento de acuerdo al ancho de banda disponible estimado y teniendo en cuenta el nivel de ocupación del *buffer* del cliente. La transmisión *streaming* se realiza mediante peticiones HTTP-GET. El proceso completo de adaptación se lleva a cabo para cada petición de segmento.

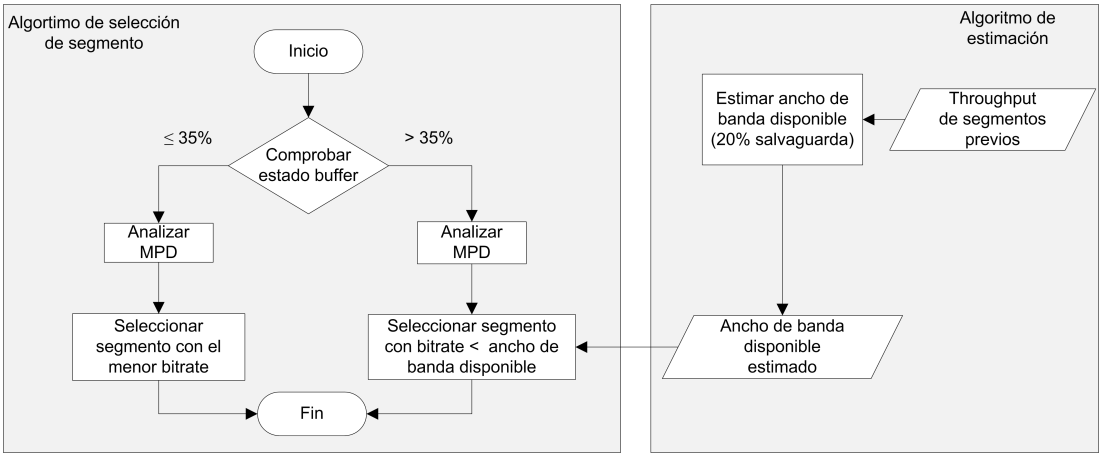


Figura 12.2: Diagrama de flujo del sistema de adaptación

En el módulo de estimación, el cliente mide el tiempo que tarda en descargar cada segmento para estimar el *throughput*, considerando la media de los últimos 5 valores (al igual que el sistemade transcodificación y el sistema adaptativo basado en SVC, presentados ambos en capítulos anteriores) y añadiendo una salvaguarda del 20 %. Este comportamiento conservador nos permite evitar situaciones no deseadas como las fluctuaciones en el ancho de banda disponible (Wu et. al, 2000). Además, de acuerdo al trabajo presentado por (Wang et. al, 2008), en el *streaming* TCP el ancho de banda disponible debe ser, al menos, el doble del *bitrate* de vídeo. Otros autores como (Biernacki and Tutschku, 2014) concluyen en su estudio que las condiciones realmente son menos restrictivas, siendo suficiente que el *throughput* de la red sea un 15 % superior a la tasa de codificación del vídeo. Nuestra propuesta es una solución de compromiso entre las dos, añadiendo una salvaguarda del 20 %.

En el algoritmo de selección de segmento el cliente tiene en consideración el nivel de ocupación en el *buffer*, expresado en términos del porcentaje del tamaño total del *buffer*, y el *bitrate* de las diferentes representaciones de los contenidos. Si el porcentaje de ocupación del *buffer* es mayor que el umbral establecido, el módulo de adaptación selecciona el siguiente segmento a solicitar de acuerdo al ancho de banda disponible. La selección debe tener en cuenta también que el *bitrate* del segmento que se seleccione debe ser menor que

el ancho de banda estimado disponible. Toda la información necesaria acerca de los detalles de los contenidos almacenados en el servidor (incluyendo el *bitrate* de las diferentes representaciones) se extrae del fichero MPD.

Si por el contrario, el nivel de ocupación del *buffer* se encuentra por debajo del umbral establecido, el algoritmo seleccionará el segmento cuyas características de *bitrate* se correspondan con el valor más bajo de entre todas las representaciones disponibles. El objetivo de esta medida cautelar es favorecer el llenado parcial del *buffer* lo más rápidamente posible, previniendo efectos poco favorables en la reproducción de los contenidos en situaciones típicas de escenarios congestionados en las que el *buffer* del cliente se vacía.

En el algoritmo de selección de segmento propuesto, el cliente solicitará la versión de los contenidos con menor calidad hasta que se alcance un umbral mínimo en el *buffer* del cliente. El valor de este umbral se establece en un 35 % de la ocupación total. Algunos autores se centran en técnicas de adaptación basadas en niveles de *buffer* como la métrica principal. El trabajo presentado por (Zhou et. al, 2012) es un ejemplo. Los autores presentan un caso de estudio con un nivel mínimo de buffer del 10 % del tamaño total. La definición de los niveles puede depender de los requerimientos específicos del servicio. En el sistema propuesto, con vídeo bajo demanda, el umbral en el *buffer* del cliente del 35 % proporciona robusted al sistema.

Los módulos propuestos de estimación y adaptación pueden integrarse facilmente en un cliente DASH. Es más, el sistema completo de adaptación presenta una solución no-intrusiva para llevar a cabo el proceso de adaptación en la arquitectura *streaming*.

12.2. Experimento 1: Evaluación del estimador basado en los niveles de *buffer* y *throughput*

En el siguiente conjunto de pruebas se procede a evaluar el sistema de adaptación presentado en la sección 12.1.

12.2.1. Definición de objetivos

Estudiaremos un servicio de vídeo *streaming* HTTP adaptativo con contenidos de alta calidad, bajo una arquitectura simple de cliente-servidor con una línea de capacidad variable, desde un punto de vista investigador, con el fin de evaluar el comportamiento del estimador.

El estimador no deberá generar situaciones de oscilación indeseadas en el bitrate de los contenidos que se reproducen. Además, la solicitud de los segmentos que realiza el cliente debe ser acorde a la capacidad disponible en la red. También tendremos en cuenta que los incrementos o decrementos en el bitrate de los segmentos solicitados deberán realizarse únicamente en los casos necesarios y en el sentido correcto, evitando realizar, por ejemplo, ajustes de incremento de tasa cuando en realidad tenemos una situación con congestión.

12.2.2. Generación de preguntas y definición de métricas

¿Cómo es el comportamiento del sistema HTTP *streaming* adaptativo ante un canal cuya capacidad varía con el tiempo? ¿el algoritmo realiza estimaciones de manera correcta? ¿en situaciones de equilibrio se vuelven a generar ajustes?

El objetivo de un sistema adaptativo es mantener la tasa a la que se sirven los contenidos ajustada al ancho de banda disponible. Las situaciones que se desean evitar varían ligeramente en el escenario que se propone, donde el protocolo de transporte subyacente es TCP. En este caso, no exisistirán pérdidas en la red, pero sí que existen otros factores no deseados que afectan a la calidad de las reproducciones.

En este contexto es preciso analizar el porcentaje total del tiempo de reproducción en el que el nivel del *buffer* del cliente se encuentra por debajo del umbral establecido. Tales circunstancias están provocadas por un consumo de material multimedia a una velocidad mayor que la velocidad de petición y descarga de los segmentos. Es un indicativo de congestión en la red.

Siguiendo con el razonamiento planteado, la situación extrema sucedería cuando el *buffer* del cliente no contenga segmentos almacenados y la reproducción se detenga. Deberemos analizar, por tanto, el porcentaje de tiempo durante la reproducción en el que no existen segmentos en el *buffer* del cliente y como consecuencia la reproducción de los contenidos se suspende momentáneamente.

Por último, y en contraposición a las métricas anteriores planteadas, deberemos evaluar el porcentaje de tiempo en el que el *buffer* del cliente está al máximo de su capacidad, siendo éste el indicativo del funcionamiento óptimo.

12.2.3. Fase de preparación del escenario y de la recogida de datos

Para evaluar el sistema de adaptación presentado en la sección 12.1, proponemos un entorno de pruebas similar al entorno presentado en los experimetos de los sistemas anteriores (basados en transcodificación y en codificadores escalables) consistente en un servidor, un cliente DASH y el emulador de red NS-3 para modificar las condiciones del ancho de banda disponible en un entorno controlado.

El cliente DASH (que incluye un cliente HTTP para realizar las peticiones GET y el reproductor) se implementa en C++ usando las librerías del software de referencia de DASH (Muller et. al, 2013). El cliente se modifica para incluir los algoritmos de estimación y adaptación presentados en la sección 12.1. El tamaño del *buffer* del cliente se establece en 30 segmentos de longitud, para propósitos de evaluación.

El servidor consta de un servidor web Apache2. Los contenidos de vídeo almacenados en el servidor web se obtuvieron del dataset de DASH publicado por (Lederer et. al, 2012). Los contenidos seleccionados de dicha base de datos se corresponden con el vídeo *Big Buck Bunny* con una resolución de 480p. Los bitrates de las diferentes representaciones abarcan desde los 100 Kbps hasta los 4.5 Mbps.

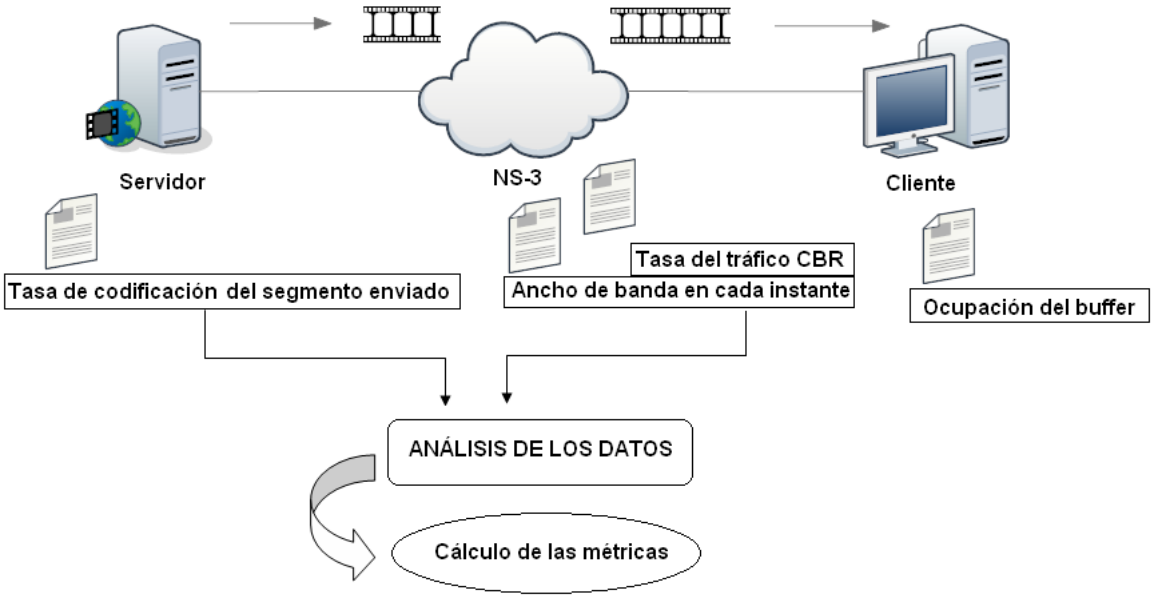


Figura 12.3: Experimento 1

En el *streaming* HTTP, el contenido se divide en segmentos de corta duración. Generalmente, el tamaño del segmento se encuentra entre 2 y 10 segudos (Begen et. al, 2011). Estudios recientes que evalúan el efecto de los diferentes tamaños de los segmentos en la transmisión streaming usando DASH concluyen que, los segmentos de tamaño pequeño optimizan el uso de la red (Lederer et. al, 2012). Por esta razón, en nuestro caso, el tamaño del segmento utilizado es de 2 segundos. Además, una corta duración del segmento permite que el proceso de adaptación sea más rápido. Puesto que nuestra estimación del *throughput* está basada en el *throughput* de segmentos previos, si la duración del segmento es larga (10 segundos) el método no es factible para reaccionar frente a fluctuaciones en el ancho de banda (Thang et. al, 2012).

El comportamiento del sistema propuesto se analiza en diferentes contextos que incluyen escenarios congestionados y no congestionados. Los patrones de variación del ancho de banda disponible se muestran en las figuras de la sección 12.2.6. Para modificar las condiciones de la red hemos empleado la herramienta de emulación NS-3, con tráfico *background* CBR para congestionar el enlace. El ancho de banda disponible resultante varía de 0.3 Mbps a 6 Mbps.

Para obtener resultados estables, llevamos a cabo experimentos de larga duración, con 10 repeticiones cada uno.

Los datos necesarios para el cálculo de las métricas de evaluación se extraen de las aplicaciones durante la ejecución del experimento y se almacenan en ficheros de texto para su posterior análisis (Figura 12.3).

12.2.4. Realización del experimento

Los detalles de la arquitectura y el esquema empleado para llevar a cabo todas las pruebas con los diferentes escenarios pertenecientes a este experimento pueden encontrarse en el apéndice D.

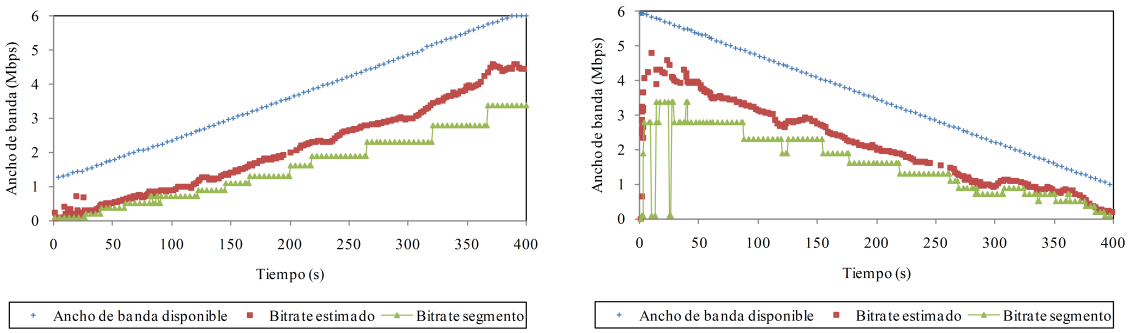
12.2.5. Recogida y validación de los datos

Se analizan los datos capturados para proceder al cálculo de las métricas propuestas.

12.2.6. Extracción de conclusiones

Como hemos descrito en la Sección 12.1, el cliente analiza el fichero MPD y comienza a solicitar los contenidos que tienen la menor calidad. Una vez que el *buffer* alcanza el nivel mínimo establecido, comienza la fase de monitorización del ancho de banda disponible y el cliente analiza los valores de las métricas para estimar el ancho de banda disponible. Si hay suficientes recursos libres en la red, el cliente solicitará el siguiente segmento con un *bitrate* mayor.

Las Figuras 12.4 a 12.5 muestran algunos ejemplos de los resultados gráficos de los procesos de estimación y adaptación en los escenarios evaluados. Las Figuras 12.4a y 12.4b muestran que, en los casos en los que existen cambios, tanto en sentido ascendente como en sentido descendente en el ancho de banda disponible, el algoritmo no realiza sobreestimaciones de la capacidad disponible en la red, de manera que el cliente no solicita segmentos que tengan un *bitrate* mayor que el ancho de banda disponible.

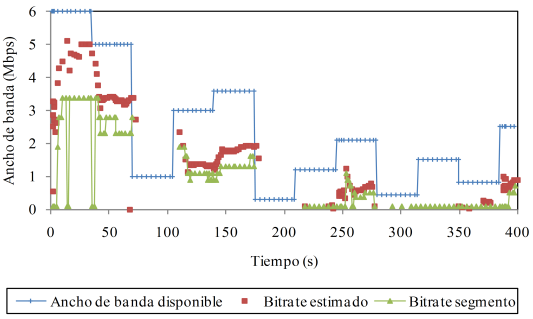


(a) Rampa ascendente

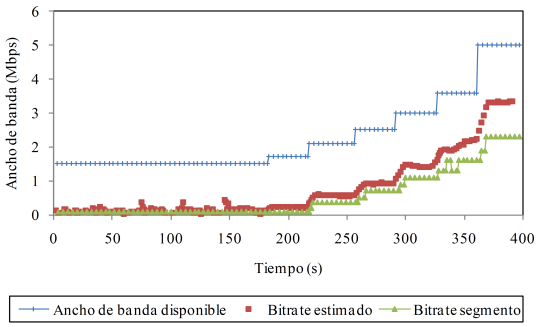
(b) Rampa descendente

Figura 12.4: Modelos rampa

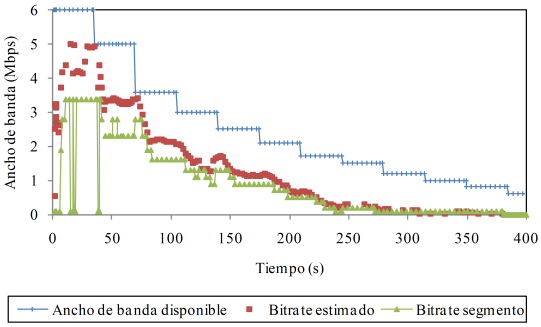
Las Figuras 12.4b, 12.5a y 12.5c muestran el comportamiento del sistema cuando el ancho de banda disponible presenta valores elevados en los instantes iniciales de la transmisión. En algunos casos, el cliente solicita el segmento con el menor bitrate (100 Kbps) aunque el algoritmo haya estimado correctamente el ancho de banda disponible. Esto es porque, al inicio de la transmisión, los niveles de *buffer* presentan valores por debajo del umbral mínimo establecido, por lo que, a pesar de contar con recursos libres en la red, el sistema permanece en la fase de llenado de *buffer* (o *rebuffering*).



(a) Escalón aleatorio



(b) Escalón ascendente



(c) Escalón descendente

Figura 12.5: Modelo escalón

La Figura 12.5a muestra los instantes en los que hay un descenso brusco en el ancho de banda disponible. Puede ocurrir que el cliente ya hubiese comenzado la descarga del segmento con un *bitrate* mayor que el ancho de banda que hay disponible en el instante actual. La congestión provocará que la descarga del segmento ocupe más tiempo. La situación se normaliza cuando el cliente solicita el siguiente segmento con el *bitrate* apropiado.

Para evaluar la estabilidad en el proceso de estimación, proponemos los escenarios de las Figuras 12.5a, 12.5b y 12.5c con modelos escalonados. Después de la fase inicial, mientras el ancho de banda permanezca constante, el algoritmo mantiene el *bitrate* del segmento. Esta es una características importante para un algoritmo adaptativo, puesto que algunos trabajos relacionados con evaluaciones subjetivas concluyen que los cambios constantes e innecesarios en el *bitrate* afectan negativamente a la calidad percibida por el usuario (Mok et. al, 2012).

Puesto que el reproductor de vídeo elige un nivel de calidad menor que el valor del ancho de banda estimado, la tasa de descarga será mayor que la tasa de reproducción. De esta manera, el cliente solicita los datos multimedia que mejor se ajustan al ancho de banda disponible en la red, pero sin congestionar el medio. En algunos casos, las diferencias entre el ancho de banda disponible y el bitrate del segmento es de casi un 50 %, debido a la salvaguarda introducida por el algoritmo de estimación y a los valores discretos de los *bitrates* de los segmentos. Esto también previene eventos de *rebuffering* y vaciado del *buffer*.

Sin embargo, cuando hay una interrupción o una reducción en el bitrate, es posible que el *buffer* no satisfaga el nivel mínimo de ocupación requerido, por lo que el algoritmo

	Media	Varianza	Intervalos de confianza
Modelo rampa ascendente	3.97	0.53	[3.45 - 4.49]
Modelo rampa descendente	1.82	0.05	[1.66 - 1.97]
Modelo escalonado	55.02	23.43	[51.56 - 58.48]
Modelo escalonado ascendente	36.15	11.66	[33.26 - 39.57]
Modelo escalonado descendente	15.16	11.66	[33.26 - 39.57]

Tabla 12.1: Parámetros estadísticos del tiempo total (%) en el que la ocupación del buffer es inferior al umbral establecido

entra en fase de *rebuffering*, tal y como ya se ha explicado. Durante el *rebuffering*, la calidad se degrada a la calidad más baja disponible. De esta manera nos aseguramos que el proceso de llenado parcial del *buffer* se lleve a cabo en el periodo de tiempo más corto posible. Sin embargo, los cortes durante la reproducción de los contenidos pueden ocurrir en aquellos casos en los que el ancho de banda sea reducido debido al vaciado completo del *buffer* del cliente (Wang et. al, 2008).

La Tabla 12.1 muestra los resultados de media, varianza y el intervalo de confianza al 95 % del porcentaje de tiempo en el que el nivel de ocupación en el buffer del cliente está por debajo del umbral mínimo. Tal y como muestran los resultados numéricos, para los escenarios del modelo de rampa los valores no exceden el 4 % del tiempo total. Los modelos escalonados ascendente y descendente presentan valores mayores. Finalmente, para el modelo escalonado mixto, durante casi la mitad del tiempo total los niveles del *buffer* se sitúan por debajo del umbral mínimo, debido a las condiciones extremas relacionadas con las reducciones del ancho de banda disponible para este escenario.

La Figura 12.6 muestra la observación menor, el cuartil, mediana y cuartil superior y la mayor observación y, en algunos casos, las observaciones que presentan un valor fuera de rango. Los resultados están presentados en el mismo orden de magnitud que los resultados de la Tabla 12.1.

Para completar la interpretación de estos resultados, la Tabla 12.2 y la Figura 12.7 resumen la media, la varianza, los intervalos de confianza al 95 % y el boxplot para el tiempo total (%) en el que la reproducción de los contenidos se detiene debido al vaciado del *buffer* del cliente. Para los modelos de rampa y para el modelo escalonado ascendente, no hay cortes durante el visionado de los contenidos, puesto que el *buffer* del cliente nunca se vacía. Para el modelo escalonado descendente, se identifican algunos problemas al final de la simulación, donde hay fuertes restricciones en la disponibilidad. El peor escenario lo presenta el modelo escalonado, donde la interrupción del servicio ocurre durante largos periodos de tiempo.

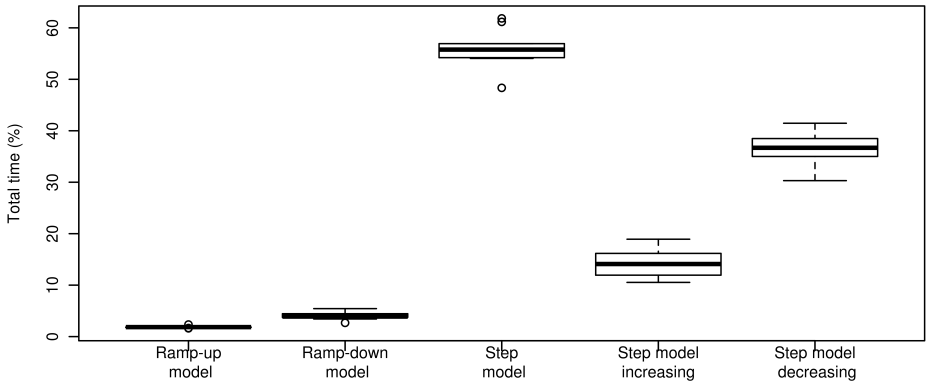


Figura 12.6: Boxplot para el tiempo total (%) en el que la ocupación del buffer es inferior al umbral establecido

	Media	Varianza	Intervalos de confianza
Modelo rampa ascendente	0	0	[NaN - NaN]
Modelo rampa descendente	0	0	[NaN - NaN]
Modelo escalonado	38.52	78.91	[32.16 - 44.87]
Modelo escalonado ascendente	0.004	0.0001	[-0.005 - 0.01]
Modelo escalonado descendente	3.34	3.70	[1.54 - 5.10]

Tabla 12.2: Parámetros estadísticos del tiempo total (%) de parada durante la reproducción

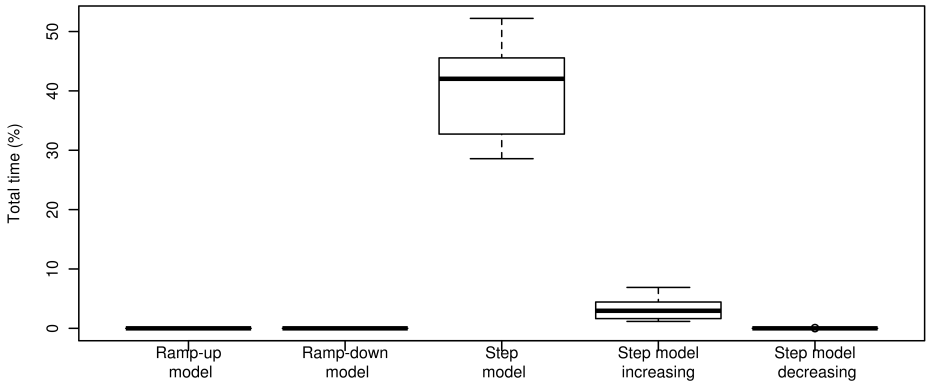


Figura 12.7: Boxplot para el tiempo total (%) de parada durante la reproducción

	Media	Varianza	Intervalos de confianza
Modelo rampa ascendente	56.46	2.37	[55.36 - 57.56]
Modelo rampa descendente	52.06	20.79	[48.79 - 55.32]
Modelo escalonado	6.63	0.84	[5.97 - 7.29]
Modelo escalonado ascendente	25.56	37.76	[21.16 - 29.95]
Modelo escalonado descendente	43.98	11.74	[41.53 - 46.44]

Tabla 12.3: Parámetros estadísticos del tiempo total (%) en el que el nivel de ocupación del buffer es del 100 %

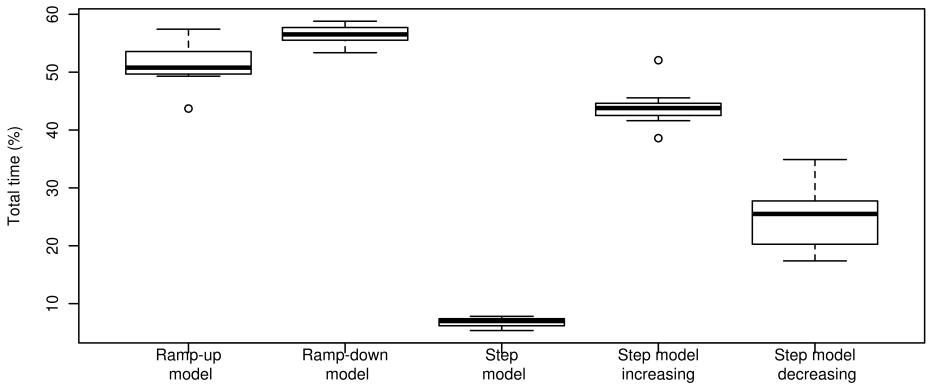


Figura 12.8: Boxplot para el tiempo total (%) en el que el nivel de ocupación del buffer es del 100 %

Por último, la Tabla 12.3 y la Figura 12.8 muestra el tiempo total (%) en el que el buffer está al 100 % de capacidad. En este sentido, para la mayoría de los escenarios evaluados el porcentaje está por encima del 25 %. En los casos en los que hay pequeños cambios en el ancho de banda disponible, estos resultados garantizan la continua reproducción de los contenidos. Por estas razones, el modelo escalonado presenta los menores porcentajes de tiempo de total ocupación del *buffer* del cliente.

12.3. Experimento 2: comparación entre el sistema adaptativo basado en DASH y el sistema streaming RTP con SVC

En el campo de la transmisión de vídeo a través de Internet hay dos tendencias claramente diferenciadas: por un lado las soluciones tradicionales basadas en UDP y, por otro lado, las recientes propuestas empleando TCP como mecanismo de transporte.

Los capítulos anteriores analizaban procesos de estimación y adaptación con RTP como protocolo de transporte. En un primer enfoque, la adaptación de los contenidos se llevaba a cabo mediante la transcodificación en tiempo real (capítulo 10). Las mejoras aplicadas al sistema inicial se centraron en el uso de la tecnología SVC, evitando así la sobrecarga en el servidor (capítulo 11). Ambas propuestas emplean paquetes de *feedback* RTCP para proporcionar información acerca del estado de la transmisión del cliente al servidor. Las métricas para estimar el ancho de banda disponible están basadas en el ratio de paquetes perdidos, el *jitter* o la linealidad de los tiempos de recepción de los paquetes RTP.

Una ventaja de SVC es la posibilidad de distribuir información en múltiples capas con la mínima redundancia. Esto hace que sea eficiente en términos de almacenamiento con varios niveles de calidad. La desventaja de SVC es la complejidad en la generación de los *streams* de vídeo y las restricciones que impone el codificador, por lo que la adopción de SVC ha sido relativamente lenta. Los resultados de adaptación del sistema propuesto basado en SVC concluyen que la selección de las capas que se transmiten se realiza de manera rápida y precisa.

En términos de escalabildiad, la solución basada en SVC presenta mejores resultados que la opción de transcodificación. Sin embargo, todavía existen limitaciones en este campo, debido a que el servidor necesita mantener una sesión de control para cada cliente. Además, las redes actuales están construidas empleando CDNs y *firewalls*, muchas de las cuales no soportan el *streaming* RTP.

HTTP evita este tipo de problemas. En este capítulo, la propuesta de estimación y adaptación se lleva a cabo empleando la tecnología DASH, cambiando el enfoque de los capítulos anteriores y moviendo la lógica de adaptación al cliente. En este caso, el proceso de estimación está basado en métricas distintas a las métricas empleadas para las soluciones basadas en RTP. Empleamos así el *throughput* estimado a partir de los segmentos anteriores y el nivel de ocupación en el *buffer*, como métricas en los procesos de estimación y adaptación.

Cualquiera de los casos propuestos y analizados emplean técnicas no-intrusivas para llevar a cabo la estimación del ancho de banda en el lado del cliente. En las soluciones basadas en RTP, los valores de las métricas se comunican al servidor a través de paquetes RTCP. En la solución basada en DASH, el cliente usa la información de las métricas para tomar las decisiones acerca del cambio de *bitrate* del *stream* solicitado.

Ambas tecnologías (las soluciones basadas en UDP/RTP y en TCP) son totalmente diferentes y deben ser comparadas desde un punto de vista de calidad percibida. En la Tabla 12.4 se resumen las características de cada uno de los sistemas.

	Sistema basado en SVC	Sistema basado en DASH
Protocolo de transporte	RTP/UDP	HTTP/TCP
Lógica de adaptación	En el servidor	En el cliente
Mecanismo de feedback	RTCP	—
Métricas de estimación	Paquetes perdidos, jitter, linealidad en los instantes de recepción de los paquetes RTP	Througput, nivel del buffer del cliente

Tabla 12.4: Características de los sistemas evaluados

12.3.1. Definición de objetivos

Estudiaremos la calidad del vídeo recibido en un sistema basado en codificación escalable SVC frente a un sistema basado en HTTP *streaming*.

12.3.2. Generación de preguntas y definición de métricas

Puesto que ambos sistemas presentan características diferentes, la comparación únicamente será posible desde el punto de vista de calidad de vídeo recibida. En este sentido, emplearemos la métrica de la PSNR, tal y como se detalla en la sección 9.2.2.

12.3.3. Fase de preparación del escenario y de la recogida de datos

Para realizar la comparación de los dos sistemas, hemos usado los mismos escenarios de evaluación que los usados en el sistema basado en SVC (ver Sección 11.3.3). La red de acceso se configura con una capacidad de 10 Mbps y el tráfico CBR sigue diferentes modelos de tendencia, con un máximo valor de 6Mbps, por lo que el ancho de banda disponible variará entre 4Mbps y 10Mbps.

La secuencia empleada en la evaluación es el vídeo denominado «BRIDGE», obtenido de los repositorios de las secuencias de test de vídeo[1] y con una resolución CIF. Las características del vídeo codificado mediante SVC se pueden ver en la Tabla 12.5 y se corresponden con las mismas características que el vídeo empleado en la evaluación de la sección 11.3. El contenido de vídeo codificado incluye dos tipos de escalabilidad SVC: temporal y de calidad. En términos de escalabilidad temporal el vídeo contiene dos capas: T0 y T1 con valores de 15 fps y 30 fps respectivamente. Además, existen cuatro niveles extra de calidad (desde Q0 a Q3) para cada una de las capas temporales. Por otro lado, hemos utilizado las herramientas de codificación disponibles para DASH (DASHEncoder

[1] http://trace.eas.asu.edu

(Lederer et. al, 2012)) para codificar los mismos contenidos de vídeo y con el mismo *bitrate*, pero adaptando las características a la tecnología DASH.

	T0	T1
Q0	2.9Mbps	4Mbps
Q1	3.6Mbps	5.4Mbps
Q2	4.2Mbps	6.5Mbps
Q3	5.3Mbps	8.6Mbps

Tabla 12.5: Propiedades del vídeo SVC empleado en el experimento

Para el cálculo de la métrica de evaluación deberemos almacenar los contenidos multimedia recibidos en el cliente.

12.3.4. Realización del experimento

Los detalles de la arquitectura y el esquema empleado para llevar a cabo todas las pruebas con los diferentes escenarios pertenecientes a este experimento pueden encontrarse en el apéndice D.

12.3.5. Recogida y validación de los datos

Se analizan los datos capturados para proceder al cálculo de las métricas propuestas.

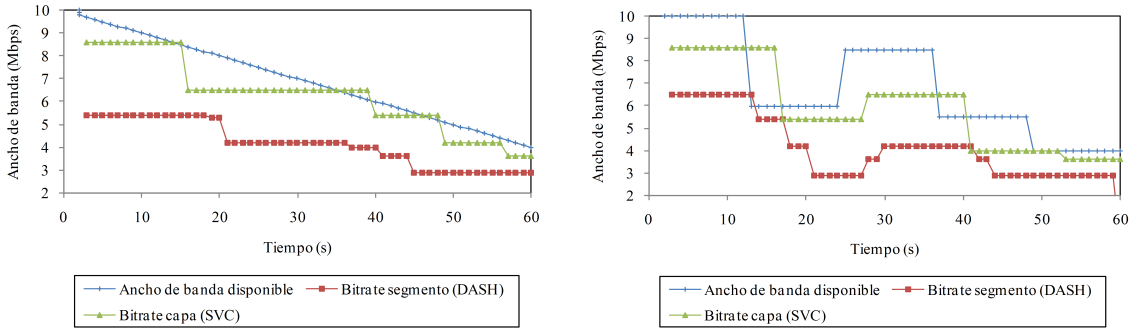
12.3.6. Extracción de conclusiones

Las Figuras 12.9a y 12.9b muestran un ejemplo del proceso de adaptación para el mismo escenario en un sistema basado en SVC y en el sistema DASH. Los ejemplos gráficos muestran el diferente comportamiento de cada algoritmo de adaptación.

En el sistema basado en SVC, el servidor lleva a cabo el proceso de adaptación a partir de la información que contienen los paquetes RTCP-APP que recibe del cliente. El cliente envía las métricas para estimar el ancho de banda disponible en los paquetes RTCP-APP cada 5 segundos. Por tanto, las sobreestimaciones pueden ocurrir en los instantes en los que la capacidad de la red disminuye y las métricas aún no han sido actualizadas.

Los módulos de estimación y adaptación propuestos para DASH desatan eventos de estimaciones del ancho de banda disponible para cada petición de segmento. Por tanto, los cambios en la capacidad de la red se detectan primero. Sin embargo, el uso de los recursos libres no es tan eficiente como en los escenarios RTP: el streaming TCP requiere más ancho de banda disponible para transmitir correctamente los mismos contenidos. A esto hay que añadir que el sistema diseñado incluye una salvaguarda del 20 % sobre el ancho de banda estimado. Como resultado, el *bitrate* que se selecciona en el sistema DASH siempre es menor (Figuras 12.9a y 12.9b).

Los sistemas adaptativos también tienen que tener en consideración la calidad percibida. Los autores (Alvarez et. al, 2013) han llevado a cabo experimentos subjetivos para



(a) Modelo rampa descendente (b) Modelo escalón

Figura 12.9: Comparación de los sistemas evaluados

evaluar las preferencias hacia un sistema adaptativo similar al presentado en el capítulo 11. Los resultados muestran que el sistema adaptativo mejora la QoE en la mayoría de las situaciones. Además, los usuarios prefieren una reducción en calidad antes que la reducción en las capas temporales. En este sentido, el sistema DASH propuesto únicamente implementa reducción de calidad a la hora de llevar a acabo el proceso de adaptación.

Hay diferentes métricas que estiman la calidad subjetiva esperada. La métrica más extendida para evaluar la calidad de vídeo es la PSNR (*Peak Signal-to-Noise Ratio*). Hemos hecho uso de esta métrica de calidad de vídeo para estudiar la transmisión del contenido multimedia desde una perspectiva *end-to-end*. La Tabla 12.6 resume los resultados del valor de la métrica de la PSNR para cada escenario. Hemos incluido dos nuevos modelos (modelos triangulares) para reproducir todas las condiciones evaluadas en la sección 11.3.3. Los resultados muestran que el sistema de estimación y adaptación propuesto basado en DASH se comporta mejor en términos de calidad de vídeo recibido comparado con el sistema basado en SVC y, en algunos casos, la mejora obtenida está por encima de los 6dB. Puesto que el sistema basado en SVC puede introducir paquetes perdidos a consecuencia de emplear UDP en la capa de transporte, la métrica de la PSNR puede presentar valores más bajos a pesar de que el *bitrate* de transmisión sea mayor.

Sin embargo, a pesar del hecho de que la calidad percibida por el usuario se ve mejorada ligeramente, otra cuestión a considerar son las sobrecargas que introducen TCP y HTTP debido a las cabeceras de los paquetes del protocolo. Comparado con RTP, HTTP impone mucha más sobrecarga (Lohmar et. al, 2011). Nuestros resultados muestran que, bajo las mismas condiciones, el sistema basado en DASH incrementa el tráfico total en un 2.91 %. Es más, tal y como ya se ha explicado, la utilización del ancho de banda en las redes congestionadas no es tan bueno como la del sistema basado en SVC. Por tanto, los sistemas adaptativos SVC y DASH tienen ventajas y desventajas que los administradores de la red tienen que tener en consideración a la hora de desplegar un servicio u otro.

12.4. Conclusiones finales

A pesar de que el sistema adaptativo basado en SVC, propuesto en el capítulo anterior, muestra un comportamiento adecuado, tanto a nivel de ajuste de los contenidos a las

	PSNR para el sistema SVC	PSNR para el sistema DASH
Modelo tendencia ascendente	39.9105	40.4593
Modelo tendencia descendente	43.3090	41.3952
Modelo escalonado	40.2943	41.2373
Modelo escalonado ascendente	36.7324	40.3380
Modelo escalonado descendente	41.4267	41.1949
Modelo triangular ascendente	39.6710	40.8126
Modelo triangular descendente	34.7573	40.9498

Tabla 12.6: Valores PSNR (dB) para cada sistema

condiciones de red como a nivel de escalabilidad, el principal problema de esta solución es la necesidad de una infraestructura específica para el despliegue del servicio (Wiegand et. al, 2009).

Como solución, planteamos el diseño e implementación de un sistema adaptativo empleando la tecnología DASH.

Puesto que los protocolos empleados para el transporte del contenido multimedia son totalmente diferentes a los empleados en los sistemas adaptativos vistos en capítulos anteriores, las métricas de estimación y adaptación serán también diferentes. En este caso emplearemos la medida del *throughput* y el valor del nivel de ocupación en el *buffer* del cliente.

Las cuestiones referentes a la escalabilidad del sistema quedan totalmente resueltas debido a que en DASH, la lógica de estimación y adaptación se localiza en el cliente. Además, la calidad del vídeo recibido se ve ligeramente mejorada frente a un sistema adaptativo basado en SVC.

Parte V

Conclusiones y trabajos futuros

Capítulo 13

CONCLUSIONES

El contenido multimedia supone una importante fracción en el incremento del tráfico en Internet. De hecho, el streaming de vídeo supone una de las aplicaciones más importantes que contribuyen a esta tendencia. Hoy en día, las conexiones de banda ancha permiten que las aplicaciones multimedia alcancen resoluciones de alta definición y frame rates elevados que requieren tasas de compresión del orden de Mbps para los contenidos.

Una cuestión clave en este escenario planteado es proporcionar al usuario la mayor calidad de experiencia posible para el entorno multimedia en el que se encuentre, es decir, de acuerdo a los recursos de red disponibles. Para este propósito, la adaptación del contenido multimedia al ancho de banda disponible es crucial. El streaming de vídeo adaptativo es un avance relevante con respecto al streaming basado en descarga progresiva, empleado por conocidos portales de vídeo en Internet, y consiste en adaptar dinámicamente el bitrate del contenido con el fin de proporcionar la máxima QoE para las condiciones de ancho de banda actuales, asegurando así la continua reproducción.

Sin embargo, garantizar una cierta calidad de experiencia no está exento de problemas: errores en la transmisión, retardos o disponibilidad del ancho de banda son factores que afectan a la calidad percibida en un servicio de vídeo streaming. Por tanto, el streaming adaptativo sobre redes IP es altamente recomendado para garantizar una cierta calidad de experiencia bajo condiciones cambiantes en el ancho de banda disponible.

En los primeros capítulos de la memoria, Parte I, hemos comentado la importante evolución que están teniendo los servicios de vídeo en Internet. Cada vez son más los usuarios que demandan este tipo de recursos. Sin embargo, al tratarse de servicios con requerimientos de tiempo real, son muy sensibles a las variaciones de las condiciones de red, necesitando entornos estables para la transmisión.

Por otro lado, las capacidades de las líneas de acceso de los usuarios han aumentado considerablemente. De la misma manera, el tráfico que generan los usuarios también se ha visto incrementado, por lo que las situaciones de congestión, competencia de recursos, pérdidas y errores son probables. En estos casos, cuando la ocupación de la línea es elevada, la transmisión de vídeo tradicional sufre una degradación de la QoE (*Quality of Experience*). En los experimentos hemos comprobado que en un sistema de transmisión de audio/vídeo tradicional que no cuenta con herramientas de adaptación sufre una de-

gradación de la QoE cuando la ocupación de la línea es elevada y se producen pérdidas de paquetes. Es por eso que se hace necesario contar con un sistema que permita adaptar el formato de los contenidos a las condiciones de transmisión de manera que se eviten las pérdidas de paquetes que provocan cortes en la reproducción y disminuyen notablemente la experiencia de usuario.

En base a esto hemos diseñado e implementado tres sistemas de estimación y adaptación para tres tecnologías diferentes de adaptación, empleando técnicas de estimación no instrusivas. Los mecanismos de adaptación empleados pretenden cubrir diversas necesidades, empleando para ello diferentes mecanismos de transmisión (UDP o TCP) y diferentes técnicas de adaptación: la transcodificación de los contenidos, el empleo de codificadores escalables y la disponibilidad de múltiples versiones de los contenidos en el servidor.

Como solución al problema planteado, hemos diseñado, primeramente, un un estimador no intrusivo para servicios de vídeo *streaming* basado en técnicas de transcodificación, Capítulo 10. En este sistema, la aplicación en el cliente analiza los paquetes de vídeo que recibe. Esos datos se envían al servidor, que en base a esa información es capaz de seleccionar la tasa de transmisión que mejor se ajusta al ancho de banda disponible. La adaptación se realiza tanto si el ancho de banda disponible aumenta o disminuye, ajustando el formato de los contenidos, evitando que se produzcan pérdidas y mejorando la QoE.

Decimos que es un estimador no intrusivo porque los resultados muestran que la sobrecarga que se introduce en la red para poder implementar el sistema de adaptación no supera el 0.014 %, por lo que el consumo extra de recursos de red es mínimo. Esta es una ventaja clara frente a otras técnicas analizadas en el capítulo 8, cuyo principal inconveniente era la necesidad de inyectar tráfico adicional para realizar la estimación, lo que podía acarrear una saturación de los recursos cuando las líneas están altamente cargadas.

Otra tarea clave se corresponde con la elección de las métricas a partir de las cuales se basará el proceso de estimación. Para ello fue necesario realizar un estudio en profundidad del tráfico *streaming* y determinar qué parámetros presentaban una relación directa con la congestión en la red.

A partir de esas métricas se estudian mejoras para el algoritmo de estimación, con el fin de encontrar el estimador que mejor se comporte de forma genérica. A lo largo de los experimentos hemos comprobado diversas configuraciones y métricas, hasta obtener unos resultados óptimos y precisos, mejorando la fiabilidad y las prestaciones de los sistemas de adaptación de trabajos previos. Cabe destacar que, analizando el comportamiento de un segundo sistema propuesto basado en difentes métricas, obtuvimos resultados muy estables, ya que conseguimos evitar las oscilaciones en la tasa de codificación. El sistema reacciona de manera rápida ante los cambios que se producen en la capacidad disponible en la red, ajustando la tasa de manera correcta.

Además hemos extendido el estudio a situaciones en las que el vídeo tiene que competir con otro tráfico en la red, demostrando que la adaptación se sigue realizando en función del ancho de banda disponible.

Por otro lado, también analizamos una arquitectura con varios clientes, comprobando que el sistema se adapta a las condiciones de red de cada cliente. Sin embargo, el punto débil del sistema se encuentra en la escalabilidad, ya que el consumo de recursos en el servidor a causa de los procesos de transcodificación limitan el número de usuarios concurrentes accediendo al servicio. Sin embargo, el objetivo de nuestro sistema no es la escalabilidad, si no la adaptación.

La evaluación del sistema y los experimentos se llevaron a cabo en una arquitectura real de cliente-servidor, en la que se integraron técnicas de emulación para poder estudiar diferentes condiciones de red sin necesidad de llegar a desplegarlas físicamente.

Para extraer los datos necesarios para los análisis efectuados, fue necesario realizar multitud de experimentos de laboratorio. Con el fin de presentar los datos de una manera clara y ordenada para el lector, así como para facilitar la realización y análisis de los mismos y reproducciones de los experimentos para trabajos futuros, se ha recurrido al método GQM.

Tras un análisis en detalle concluimos que un sistema de adaptación para servicios *streaming* con indicadores de congestión como el porcentaje de pérdidas, el *jitter* y la linealidad de los paquetes en recepción es la mejor opción para solucionar los problemas que presenta este servicio ante las variaciones de capacidad del canal. Adoptando estas medidas se mejora notablemente la QoE respecto a sistemas no adaptativos, que será el factor clave que caracterice la calidad del servicio ofrecido.

En base a las primeras conclusiones, hemos empleado las ideas vistas en el Capítulo 10 de un estimador no intrusivo basado en trascodificación para desarrollar un segundo sistema *streaming* adaptativo empleando la tecnología SVC, Capítulo 11.

Mediante las pruebas de emulación, se ha demostrado que el segundo sistema implementado es capaz de realizar una adaptación rápida y precisa ante los cambios que se producen en la capacidad disponible en la red, evitando la proliferación de indeseadas oscilaciones en los ajustes de tasas. La utilización de un esquema SVC ha permitido incrementar la escalabilidad del conjunto del sistema con respecto al diseño previo basado en transcodificación. También hemos demostrado que la adaptación se realiza de forma correcta en presencia de otro tráfico en la red o en arquitecturas con varios clientes. Además, el sistema de estimación y adaptación basado en SVC es tan escalable como un sistema no adaptativo empleando la tecnología SVC.

Por último, al igual que el sistema previo, las métricas empleadas para la estimación de los recursos disponibles constituyen un aporte novedoso al incluir la linealidad de los instantes de recepción de los paquetes RTP y la sobrecarga introducida en la red para poder implementar el sistema de adaptación no supera el 0.1 % en este caso. Además, las soluciones adoptadas permiten dotar al servidor de una buena capacidad de adaptación y el procesamiento en el lado del cliente es muy bajo.

Para finalizar, en el capítulo 12 dejamos de lado los sistemas *streaming* tradicionales,

que emplean protocolos diseñados exclusivamente para el envío multimedia en Internet, y nos centramos en protocolos, aparentemente no factibles para este tipo de tráfico, pero que están ampliamente instaurados en Internet. Hablamos del protocolo HTTP que, con el protocolo DASH proporcina la funcionalidad deseada.

El streaming basado en HTTP sigue un modelo ligeramente diferente al streaming basado en RTP. En éste último, el servidor envía los datos al cliente a un determinado bitrate hasta que finalice la sesión o, en el caso de los sistemas adaptativos propuestos, hasta que el sistema de estimación y adaptación implementados en el servidor decidan modificar la tasa a la que se envían los contenidos (modelo PUSH de protocolo *streaming*). Por contra, en el streaming basado en HTTP, es el cliente el que solicita los contenidos periódicamente al servidor, siguiendo el modelo PULL del protocolo.

Una de las ventajas del sistema adaptativo basado en DASH es la simplificación del servidor, puesto que toda la lógica de estimación y adaptación se traslada al cliente. En cuanto al diseño del algoritmo de estimación y adaptación, la sencillez es su característica principal. Las métricas empleadas para la estimación y adaptación permiten un correcto funcionamiento del sistema, sin realizar sobreestimaciones del ancho de banda disponible. Comparado con la propuesta basada en SVC, el sistema basado en DASH realiza cambios en la adaptación más precisos y rápidos.

Las pequeñas interrupciones o reducciones del ancho de banda no afectan a la calidad percibida puesto que los niveles de ocupación del buffer del cliente permanecen estables. Es más, los resultados demuestran que, en la mayoría de las situaciones evaluadas, la calidad percibida por el usuario es superior que en el sistema basado en SVC.

El mayor problema al que se enfrenta esta propuesta está relacionado con las pausas durante la reproducción en instantes en los que el ancho de banda sufre un decremento drástico. También detectamos que los recursos de red se emplean de una manera menos eficiente que en RTP y el tráfico generado se incrementa alrededor de un 3 %.

En conclusión, los estudios realizados y los diseños, implementaciones y mejoras aplicadas en los sistemas de vídeo adaptativo tienen una aplicación práctica muy importante en sectores relacionados con la distribución de contenidos multimedia de alta calidad, ya que permite dotar de cierta QoS a los servicios de vídeo *streaming*, adaptando la tasa de transferencia a la disponibilidad de ancho de banda en la red de acceso del usuario, donde las condiciones del canal son cambiantes y los recursos son limitados.

Capítulo 14

TRABAJOS FUTUROS

El trabajo desarrollado en esta tesis doctoral no supone un punto y final en el ámbito de los sistemas *streaming* adaptativos. Existen aún diversas cuestiones que pueden mejorar los prototipos propuestos y tecnologías emergentes que pueden conformar el panorama de los próximos años en la transmisión de contenidos multimedia en Internet.

El sistema de estimación de los recursos libres en la red es uno de los puntos a tratar. La robustez de los mecanismos de estimación de ancho de banda puede completarse con el desarrollo de un sistema de predicción que permita determinar la tendencia de variación del canal basándose en el comportamiento previo. Cuanto mayor sea la exactitud en las estimaciones del ancho de banda, mejor será el comportamiento del sistema adaptativo.

Por otro lado, la inclusión de nuevos escenarios y patrones de variación de tráfico nos permitirán evaluar con mayor nivel de detalle los sistemas adaptativos. A medida que introduzcamos nuevos modelos de red más complejos, deberemos tener en cuenta también las arquitecturas compuestas por CDNs (*Content Distribution Networks*) con el fin de de explotar las ventajas del *streaming* HTTP en este tipo de entornos. En estos escenarios, que implementan cachés, el análisis de nuevas métricas para el sistema de estimación y adaptación puede mejorar el comportamiento del algoritmo.

Otro aspecto clave lo presentan el desarrollo de evaluaciones subjetivas de los sistemas propuestos. Mediante los sistemas adaptativos podemos reducir las pérdidas, los retardos o las interrupciones durante la reproducción de los contenidos. Sin embargo, los usuarios juegan un papel crucial en el proceso de optimización de los algoritmos. Los servicios de vídeo adaptativos están diseñados con el objetivo de mejorar la calidad de la experiencia de los usuarios y son, por tanto, los usuarios quienes pueden arrojar resultados y conclusiones acerca de las preferencias de comportamiento del sistema.

Por último, destacamos las tres tecnologías que pueden definir el futuro de los servicios de vídeo: 4K, HEVC (*High Efficiency Video Coding*) y DASH.

- 4K: presenta un nuevo estándar de resolución de imagen. Cuadriplica a la actual resolución en píxeles de la alta definición de 1080p, proporcionando imágenes más claras. Evidentemente, el ancho de banda necesario para transmitir este tipo de contenidos supone incrementar considerablemente los costes asociados a la infraestructura de red. En este sentido, en los próximos meses nos encontraremos con

un buen número de trabajos basados en encontrar optimizaciones para el códec H.264 que permitan codificar y comprimir el contenido en esta resolución para su transmisión a través de la infraestructura de alta definición actual.

Desde el punto de vista del usuario, la incorporación del estándar 4K supone una mejora en el visionado de la imagen, reduciendo la visibilidad de la estructura de píxeles en la imagen y aumentando la nitidez en soportes visuales o pantallas grandes.

- HEVC: también denominado H.265, supone la siguiente generación de codificadores al estándar H.264 AVC. Los niveles de compresión que se pueden alcanzar con HEVC llegan incluso a duplicar el nivel de compresión obtenido con AVC para una determinada calidad de vídeo (Ohm et. al, 2012). Para el campo de la distribución de contenidos multimedia, el códec HEVC se impone como un importante hito y los fabricantes de hardware intentarán introducir rápidamente equipos compatibles con HEVC.

  La resolución 4K y el códec HEVC vendrán de la mano para cubrir las exigencias de calidad de los usuarios en una red con reducido ancho de banda. Los proveedores de contenidos también se verán beneficiados por esta tecnología, ahorrando costes de almacenamiento para los contenidos y pudiendo hacer frente a la demanda de mayor calidad de experiencia, por lo que su adopción resultará inminente en los próximos años. Pero para que este escenario sea una realidad, las aplicaciones, los formatos de los ficheros, los protocolos y los servidores *streaming* y dispositivos finales deben implementar el soporte para el nuevo codificador. Es por eso que la mayoría de las predicciones sitúan a AVC como estándar de codificación de facto hasta el año 2018, pero en el ámbito de la investigación ya comienzan a surgir los primeros trabajos, relacionados, principalmente, con la eficiencia de codificación y mejoras en el codificador.

- DASH: la industria emplea en gran medida el protocolo HTTP como el protocolo base para el desarrollo de nuevos servicios y aplicaciones debido a las ventajas que ofrecen sus características para desenvolverse en la arquitectura de red moderna que impera en Internet. El uso de cachés, proxies o las redes de distribución de contenidos constituyen la superioridad del uso de HTTP *streaming* frente al *streaming* RTP. A ello deberemos añadir la interoperabilidad que proporciona DASH, al tratarse de un estándar abierto.

  Algunas compañías de vídeo bajo demanda ya comienzan a implementar esta tecnología en sus dispositivos y la tendencia a su uso sigue en alza. Por tanto, DASH constituirá uno de los protocolos estrella en el futuro de la transmisión *streaming* en Internet.

Capítulo 15

PUBLICACIONES DERIVADAS DE LA TESIS

Fruto del trabajo de investigación realizado se desarrollan una serie de publicaciones científicas, tanto en revistas indexadas en el *Journal Citation Report* (JCR) como en congresos.

El primer prototipo desarrollado, expuesto en el capítulo 10, fue publicado en la revista Multimedia Tools and Applications. En este artículo se resume el diseño del estimador para sistemas streaming adaptativos basados en la técnica de transcodificación. La referencia completa al artículo y el abstract del mismo se detallan a continuación:

> Fraga, A., Pozueco, L., Pañeda, X.G., García, R., Melendi, D., Cabrero, S. A non-intrusive estimation for high-quality Internet TV services. *Multimedia Tools and Applications*, vol 54(3), pp.569-588, 2011. doi:10.1007/s11042-010-0566-3
>
> **Abstract**: This paper presents a non-intrusive estimator for Internet TV services based on streaming technology. Analyzing the video packets received by the client application, the estimator is capable of selecting the most suitable encoding bitrate for the available bandwidth in the end-to-end path. The estimator has been integrated in a real client/server architecture and evaluated with different network traffic situations. The results of the performed evaluation have revealed the stability and adaptation speed as the best qualities of our proposal.

A continuación, los estudios preliminares y los primeros resultados obtenidos en los sistemas adaptativos con SVC fueron presentados en las *X Jornadas de Ingeniería Telemática*.

> Pozueco, L., Pañeda, X.G., Alvarez, A., Cabrero, S., Melendi, D., García, R. Sistema de distribución de vídeo streaming adaptativo basado en codificación SVC. *X Jornadas de Ingeniería Telemática*, Cantabria, España, 2011.
>
> **Abstract**: Hoy en día, el acceso a contenidos multimedia puede realizarse a través de diversos tipos de redes y con una amplia variedad de terminales que presentan diferentes restricciones. Por tanto, la adaptación de los contenidos

a un entorno heterogéneo como es Internet será un proceso clave en la mejora de la percepción de la calidad del usuario. En este trabajo presentamos un sistema adaptativo de streaming empleando la tecnología Scalable Video Coding (SVC), recientemente estandarizada como extensión de H.264/A VC. Empleando información de feedback proveniente del cliente acerca del estado de la transmisión, el servidor es capaz de adaptar las capas que se envían del vídeo escalable en función de la congestión del enlace. El sistema se implementa en equipos reales y los resultados muestran el correcto funcionamiento ante diferentes variaciones del ancho de banda disponible, así como la escalabilidad del sistema cuando al servicio acceden clientes de manera simultánea.

Las mejoras aplicadas y la evaluación completa del sistema final adaptativo con SVC, explicado en el capítulo 11, puede consultarse en la revista *Computers & Electrical Enginnering*.

Pozueco, L., Pañeda, X.G., García, R., Melendi, D., Cabrero, S. Adaptable system based on Scalable Video Coding for high-quality video service. *Computers & Electrical Enginnering,* vol 39(3), pp 775–789*,* 2013. doi: 10.1016/j.compeleceng.2013.01.015

Abstract: Content adaptation to a heterogeneous environment like the Internet is a key process for improving the perceived quality of the user. This paper presents an adaptive streaming sys- tem using Scalable Video Coding (SVC) technology. Using feedback information from cli- ents about the transmission status, the server is able to select the most suitable combination of SVC layers for the available bandwidth. The estimation of the available bandwidth is carried out with non-intrusive methods, based on classic metrics such as packet loss, jitter and novel metrics like the linearity of reception times of RTP packets. The system is implemented in real equipment and the results show the correct operation and the accuracy of the system when adapting to different variations of the available band- width. We also study the scalability of the system when several clients access the service simultaneously, demonstrating that our system is as scalable as a non-adaptive system with SVC.

Finalmente, la última propuesta de sistemas de estimación y adaptación basada en HTTP *streaming* (capítulo 12) ha sido recientemente publicada en *Multimedia Tools and Applications*.

Pozueco, L., Pañeda, X.G., García, R., Melendi, D., Cabrero, S., Díaz, G. Adaptation engine for a streaming service based on MPEG-DASH. Multimedia Tools and Applications. 2014. doi: 10.1007/s11042-014-2034-y

Abstract: HTTP Videostreaming has become a strong candidate for video transmission on the Internet thanks to the abundance of web infrastructure. With the recent standardization of the new MPEG Dynamic Adaptive Streaming over HTTP (DASH), the flexibility and implanta- tion of adaptive video systems has increased due to the fact that DASH can operate on a conventional web infrastructure. In this paper we propose an estimation and adaptation system for DASH. The proposed adaptive algorithm is based on client buffer

threshold and smooth throughput measures (based on the throughput of previous segments). Before DASH, the standard of Scalable Video Coding (SVC) also arose from the idea of adaptation. Both systems (adaptive system based on SVC and the proposed system for DASH) are compared in terms of Video Quality (VQ) metrics. The results show that the proposed system reacts properly to changes in the network capacity, while maintaining a minimum level of segments in the buffer. The user-perceived quality is better than in the SVC-based adaptive system although the generated traffic is higher.

Paralelamente al trabajo de investigación desarrollado en esta tesis doctoral, surgen otras publicaciones en el campo de la calidad de vídeo, referentes tanto a métricas objetivas como a métricas subjetivas. Estos trabajos no están relacionados con las aportaciones de esta tesis doctoral, pero sí resultan interesantes para analizar la calidad de la experiencia de sistemas de vídeo *streaming* adaptativos. Así por ejemplo podemos citar los siguientes trabajos:

Álvarez, A., Pozueco, L., Cabrero, S., Pañeda, X.G., García, R., Melendi, D., Díaz, G. Evaluaciones subjetivas de servicios streaming adaptativos vs no-adaptativos. *XI Jornadas de Ingeniería Telemática*, Granada, España 2013.

Abstract: El análisis subjetivo de sistemas streaming adaptativos es un proceso clave para optimizar algoritmos que permitan una adaptación basada en medidas de calidad de la experiencia (QoE). En este trabajo abordamos el estudio comparativo de un sistema streaming adaptativo frente a un sistema streaming tradicional (no adaptativo) desde el punto de vista subjetivo. Se evaluará en qué situaciones y condiciones un sistema adaptativo mejora la calidad percibida por el usuario y cuáles son los umbrales de tolerancia a pérdidas que marquen el inicio del proceso de adaptación. Para ello se plantean diferentes situaciones de disponibilidad de ancho de banda en la red y diferentes decisiones de adaptación. Los resultados del estudio, en el que han participado 75 usuarios, resuelven cuestiones clave para el diseño de sistemas adaptativos y muestran que la adaptación mejora la experiencia de usuario en la mayoría de las condiciones evaluadas.

Álvarez, A., Pozueco, L., Cabrero, S., Pañeda, X.G., García, R., Melendi, D., Díaz, G. Subjective evaluation of critical success factors for a QoE aware system. *Computer Communications*, vol 36(15-16), pp 1608-1620, 2013. doi: 10.1016/j.comcom.2013.07.005

Abstract: For many years video content delivery has established itself as the killer application. Improving QoE on adaptive streaming is focusing many efforts in the quest for optimized methods and metrics to allow a QoE driven adaptation. Questions such as whether adaptive systems based on Scalable Video Coding improve subjective quality and in which situations or to what degree are still open issues. Tolerance and indifference thresholds for each type of content, conditions or viewer category, with regard to adap- tive systems are critical success factors that are yet unresolved. We compare the performance of a com- plete adaptive system with the traditional, i.e. non-adaptive,

approach in subjective terms. Results of surveying 75 participants show that the adaptation improves QoE under most of the evaluated condi- tions. Tolerance thresholds for triggering adaptation events have been identified. Users accustomed to Internet video are more critical than users that only watch TV. The under 35 year old subset among the available population is generally more satisfied with the adaptive system than the older subset.

Álvarez, A., Pozueco, L., Cabrero, S., Pañeda, X.G., García, R., Melendi, D., Díaz, G. A Framework to Measure and Estimate Video Quality in SVC Real-Time Adaptive Systems. *International Journal of Business Data Communications and Networking*, vol 10(1), pp 47-64, 2014.

Abstract: Effectively adapting the content to network conditions in real-time is an important matter in best-effort networks like the Internet. Scalable Video Coding (SVC) is an interesting alternative to implement such sys- tems. However, some problems of the performance evaluation of SVC based adaptive systems have not been solved. The authors review the main efforts directed to measure video quality on SVC related systems and discuss the limitations of each one. This paper elaborates a framework to measure video quality metrics in real adaptive SVC based streams. An estimation method for full reference video quality metrics is proposed. This method reduces reference information required and it is able to provide real-time accurate results simply using metadata regarding the video quality of the reference layers. The video quality of several streams that have been generated using a real-time adaptive system is first measured with the elaborated framework and then estimated with the proposed method.

Bibliografía

[Ahmad et. al 2007] Ahmad, Shafqaat ; Gohar, N.D. ; Kamal, Aatif: A Dynamic Congestion Control Mechanism for Real Time Streams over RTP. In: *Advanced Communication Technology, The 9th International Conference on* vol. 2, 2007, pp. 961–966. – ISBN 1738-9445

[Akhshabi et. al 2012] Akhshabi, Saamer ; Narayanaswamy, Sethumadhavan ; Begen, Ali C. ; Dovrolis, Constantine: An experimental evaluation of rate-adaptive video players over http. In: *Signal Processing: Image Communication* 27 (2012), num. 4, pp. 271–287

[Alvarez et. al 2010] Alvarez, Alberto ; Orea, Rafael ; Cabrero, Sergio ; Pañeda, Xabiel G. ; GarcÃa, Roberto ; Melendi, David: Limitations of network emulation with single-machine and distributed ns-3. In: *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques* vol. 67. ICST, Brussels, Belgium, Belgium : ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, pp. 1–9. – ACM ID: 1808228. – ISBN 978-963-9799-87-5

[Alvarez et. al 2013] Alvarez, Alberto ; Pozueco, Laura ; Cabrero, Sergio ; Pañeda, Xabiel G. ; Garcia, Roberto ; Melendi, David ; DÃaz Orueta, Gabriel: Subjective evaluation of critical success factors for a QoE aware adaptive system. In: *Computer Communications* 36 (2013), num. 15–16, pp. 1608–1620. – ISSN 0140–3664

[Amon et. al 2007] Amon, Peter ; Rathgen, Thomas ; Singer, David: File Format for Scalable Video Coding. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 17 (2007), num. 9, pp. 1174–1185. – ISSN 1051-8215

[Arsan 2008] Arsan, Taner: An Integrated Software Architecture for Bandwidth Adaptive Video Streaming. In: *Proceedings of World Academy of Science, Engineering and Technology* 27 (2008), February

[Arsan 2012] Arsan, Taner: Review of bandwidth estimation tools and application to bandwidth adaptive video streaming. In: *2012 9th International Conference on High Capacity Optical Networks and Enabling Technologies (HONET)*, 2012, pp. 152–156

[Baldo et. al 2004] Baldo, Nicola ; Horn, Uwe ; Kampmann, Markus ; Hartung, Frank: RTCP feedback based transmission rate control for 3G wireless multimedia streaming. In: *Personal, Indoor and Mobile Radio Communications, 2004. PIMRC 2004. 15th IEEE International Symposium on* vol. 3, 2004, pp. 1817–1821 Vol.3

[Basili and Weiss 1984] Basili, Victor ; Weiss, David: A Methodology for Collecting Valid Software Engineering Data. In: *IEEE Transactions on Software Engineering* SE-10 (1984), pp. 728–738. – ISSN 0098-5589

[Begen et. al 2011] Begen, Ali ; Akgul, Tankut ; Baugher, Mark: Watching Video over the Web: Part 1: Streaming Protocols. In: *IEEE Internet Computing* 15 (2011), April, num. 2, pp. 54–63. – ISSN 1089-7801

[Bianchi et. al 2009] Bianchi, Giuseppe ; Detti, Andrea ; Loreti, Pierpaolo ; Pisa, Claudio ; Proto, Francesco S. ; Kellerer, Wolfgang ; Thakolsri, Srisakul ; Widmer, Joerg: Application-aware H.264 Scalable Video Coding delivery over Wireless LAN: Experimental assessment. In: *Cross Layer Design, 2009. IWCLD '09. Second International Workshop on*, 2009, pp. 1–6

[Biernacki and Tutschku 2014] Biernacki, Arkadiusz ; Tutschku, Kurt: Performance of HTTP video streaming under different network conditions. In: *Multimedia Tools and Applications* 72 (2014), num. 2, pp. 1143–1166. – ISSN 1380-7501

[Bolot and Turletti 1998] Bolot, Jean-Chrysostome ; Turletti, Thierry: Experience with control mechanisms for packet video in the internet. In: *ACM SIGCOMM Computer Communication Review* 28 (1998), January, pp. 4–15. – ACM ID: 280551. – ISSN 0146-4833

[Bouras et. al 2008] Bouras, Christos ; Gkamas, Apostolos ; Kioumourtzis, Georgios: Smooth Multicast Congestion Control for Adaptive Multimedia Transmission. In: *Next Generation Internet Networks, 2008. NGI 2008*, 2008, pp. 146–152

[Carter and Crovella 1996] Carter, Robert L. ; Crovella, Mark E.: Measuring Bottleneck Link Speed in Packet-Switched Networks. In: *Performance Evaluation* 27 (1996), pp. 297–318

[Chang and Vetro 2005] Chang, Shih-Fu ; Vetro, Anthony: Video Adaptation: Concepts, Technologies, and Open Issues. In: *Proceedings of the IEEE* 93 (2005), num. 1, pp. 148–158. – ISSN 0018-9219

[Chen et. al 2007] Chen, Peng ; Lim, Jeongyeon ; Lee, Bumshik ; Kim, Munchurl ; Hahm, Sangjin ; Kim, Byungsun ; Lee, Keunsik ; Park, Keunsoo: A network-adaptive SVC Streaming Architecture. In: *Advanced Communication Technology, The 9th International Conference on* vol. 2, 2007, pp. 955–960. – ISBN 1738-9445

[Choi et. al 2007] Choi, Haechul ; Kang, Jung W. ; Kim, Jae-Gon: Dynamic and Interoperable Adaptation of SVC for QoS-Enabled Streaming. In: *Consumer Electronics, IEEE Transactions on* 53 (2007), num. 2, pp. 384–389. – ISSN 0098-3063

[Cisco 2014] Cisco: Cisco Visual Networking Index: Forecast and Methodology / Cisco. 2014. –

[Civanlar et. al 2001] Civanlar, M.Reha ; Luthra, Ajay ; Wenger, Stephan ; Zhu, Wenwu: Introduction to the special issue on streaming video. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 11 (2001), March, num. 3, pp. 265–268. – ISSN 1051-8215

[Cranley et. al 2006] Cranley, Nicola ; Perry, Philip ; Murphy, Liam: User Perception of Adapting Video Quality. In: *Int. J. Hum.-Comput. Stud.* 64 (2006), August, num. 8, pp. 637–647. – ISSN 1071-5819

[De Cicco and Mascolo 2014] De Cicco, Luca ; Mascolo, Saverio: An Adaptive Video Streaming Control System: Modeling, Validation, and Performance Evaluation. In: *Networking, IEEE/ACM Transactions on* 22 (2014), April, num. 2, pp. 526–539. – ISSN 1063-6692

[De Cicco et. al 2011] De Cicco, Luca ; Mascolo, Saverio ; Palmisano, Vittorio: Feedback Control for Adaptive Live Video Streaming. In: *Proceedings of the Second Annual ACM Conference on Multimedia Systems*. New York, NY, USA : ACM, 2011 (MMSys '11), pp. 145–156. – URL `http://doi.acm.org/10.1145/1943552.1943573`. – ISBN 978-1-4503-0518-1

[Downey 1999] Downey, Allen: Using pathchar to estimate Internet link characteristics. In: *Proceedings of ACM SIGCOMM*, 1999, pp. 241–250

[Eberhard et. al 2008] Eberhard, Michael ; Celetto, Luca ; Timmerer, Christian ; Quacchio, Emanuele ; Hellwagner, Hermann ; Rovati, Fabrizio S.: An interoperable streaming framework for Scalable Video Coding based on MPEG-21. In: *Visual Information Engineering, 2008. VIE 2008. 5th International Conference on*, 2008, pp. 723–728. – ISBN 0537-9989

[Ennaji et. al 2009] Ennaji, Y. ; Boulmalf, Mohammed ; Alaoui, C.: Experimental analysis of video performance over wireless local area networks. In: *Multimedia Computing and Systems, 2009. ICMCS '09. International Conference on*, April 2009, pp. 488–494

[Fraga et. al 2011] Fraga, Alberto ; Pozueco, Laura ; Pañeda, Xabiel G. ; Garcia, Roberto ; Melendi, David ; Cabrero, Sergio: A Non-intrusive Estimation for High-quality Internet TV Services. In: *Multimedia Tools and Applications* 54 (2011), sep, num. 3, pp. 569–588. – ISSN 1380-7501

[Frojdh et. al 2006] Frojdh, Per ; Horn, Uwe ; Kampmann, Markus ; Nohlgren, Anders ; Westerlund, Magnus: Adaptive streaming within the 3GPP packet-switched streaming service. In: *Network, IEEE* 20 (2006), num. 2, pp. 34–40. – ISSN 0890-8044

[Gorkemli and Tekalp 2012] Gorkemli, Burak ; Tekalp, A. M.: Adaptation strategies for MGS scalable video streaming. In: *Signal Processing: Image Communication* 27 (2012), July, num. 6, pp. 595–611. – ISSN 0923-5965

[Grafl et. al 2013] Grafl, Michael ; Timmerer, Christian ; Hellwagner, Hermann ; Xilouris, George ; Gardikis, Georgios ; Renzi, Daniele ; Battista, Stefano ; Borcoci, Eugen ; Negru, Daniel: Scalable Media Coding Enabling Content-Aware Networking. In: *IEEE MultiMedia* 20 (2013), April, num. 2, pp. 30–41. – URL `http://dx.doi.org/10.1109/MMUL.2012.57`. – ISSN 1070-986X

[Hore and Ziou 2010] Hore, Alain ; Ziou, Djemel: Image Quality Metrics: PSNR vs. SSIM. In: *Pattern Recognition (ICPR), 2010 20th International Conference on*, Aug 2010, pp. 2366–2369. – ISSN 1051-4651

[Huang et. al 2012] Huang, Te-Yuan ; Handigol, Nikhil ; Heller, Brandon ; McKeown, Nick ; Johari, Ramesh: Confused, timid, and unstable: picking a video streaming rate is hard. In: *Proceedings of the 2012 ACM conference on Internet measurement conference*, 2012, pp. 225–238

[Huynh-Thu and Ghanbari 2008] Huynh-Thu, Quan ; Ghanbari, Mohammed: Scope of validity of PSNR in image/video quality assessment. In: *Electronics Letters* 44 (2008), June, num. 13, pp. 800–801. – ISSN 0013-5194

[Huynh-Thu and Ghanbari 2012] Huynh-Thu, Quan ; Ghanbari, Mohammed: The accuracy of PSNR in predicting video quality for different video scenes and frame rates. In: *Telecommunication Systems* 49 (2012), num. 1, pp. 35–48. – ISSN 1018-4864

[ISO/IEC 2014] ISO/IEC: ISO/IEC 23009-1 Information Technology - Dynamic Adaptive Streaming over HTTP (DASH) – Part 1: Media presentation description and segment formats / International Organization for Standardization, Geneva, Switzerland. 2014. –

[Jain and Dovrolis 2002] Jain, Manish ; Dovrolis, Constantine: Pathload: A Measurement Tool for End-to-End Available Bandwidth. In: *Proceedings of Passive And Active Measurements (PAM) Workshop* (2002), pp. 14–25

[Kanakia et. al 1995] Kanakia, Hemant ; Mishra, Partho P. ; Reibman, Amy R.: An adaptive congestion control scheme for real time packet video transport. In: *Networking, IEEE/ACM Transactions on* 3 (1995), Dec, num. 6, pp. 671–682. – ISSN 1063-6692

[Kantarci 2008] Kantarci, Aylin: Streaming of scalable h.264 videos over the Internet. In: *Multimedia Tools and Applications* 36 (2008), February, num. 3, pp. 303–324. – ISSN 1380-7501, 1573-7721

[Klaue et. al 2003] Klaue, Jirka ; Rathke, Berthold ; Wolisz, Adam: EvalVid - A Framework for Video Transmission and Quality Evaluation. In: *In Proc. of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation* (2003), pp. 255—272

[Kofler et. al 2011] Kofler, Ingo ; Kuschnig, Robert ; Hellwagner, Hermann: In-network adaptation of H.264/SVC for HD video streaming over 802.11g networks. In: *Proceedings of the 21st international workshop on Network and operating systems support for digital audio and video*. New York, NY, USA : ACM, 2011 (NOSSDAV '11), pp. 9–14. – ISBN 978-1-4503-0777-2

[Kofler et. al 2012] Kofler, Ingo ; Kuschnig, Robert ; Hellwagner, Hermann: Implications of the ISO base media file format on adaptive HTTP streaming of H. 264/SVC. In: *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, 2012, pp. 549–553

[Kofler et. al 2008] Kofler, Ingo ; Prangl, Martin ; Kuschnig, Robert ; Hellwagner, Hermann: An H.264/SVC-based adaptation proxy on a WiFi router. In: *Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video*. New York, NY, USA : ACM, 2008 (NOSSDAV '08), pp. 63–68. – ACM ID: 1496061. – ISBN 978-1-60558-157-6

[Koziolek 2008] Koziolek, Heiko: Dependability Metrics. Berlin, Heidelberg : Springer-Verlag, 2008, Kap. Goal, Question, Metric, pp. 39–42. – URL http://dl.acm.org/citation.cfm?id=1806170.1806178. – ISBN 3-540-68946-X, 978-3-540-68946-1

[Kuschnig et. al 2008] Kuschnig, Robert ; Kofler, Ingo ; Ransburg, Michael ; Hellwagner, Hermann: Design options and comparison of in-network H.264/SVC adaptation. In: *Journal of Visual Communication and Image Representation* 19 (2008), December, pp. 529–542. – ACM ID: 1466321. – ISSN 1047-3203

[Lederer et. al 2012] Lederer, Stefan ; MÃŒller, Christopher ; Timmerer, Christian: Dynamic adaptive streaming over HTTP dataset. In: *Proceedings of the 3rd Multimedia Systems Conference*. New York, NY, USA : ACM, 2012 (MMSys '12), pp. 89–94. – ISBN 978-1-4503-1131-1

[Ling and ShaoWen 2009] Ling, Peng ; ShaoWen, Li: An Improved Algorithm of RTP Adaptive Transmission Control. In: *Genetic and Evolutionary Computing, 2009. WGEC '09. 3rd International Conference on*, 2009, pp. 595–599

[Liu et. al 2011] Liu, Chenghao ; Bouazizi, Imed ; Gabbouj, Moncef: Rate adaptation for adaptive HTTP streaming. In: *Proceedings of the second annual ACM conference on Multimedia systems*. New York, NY, USA : ACM, 2011 (MMSys '11), pp. 169–174. – ISBN 978-1-4503-0518-1

[Liu et. al 2012] Liu, Chenghao ; Bouazizi, Imed ; Hannuksela, Miska M. ; Gabbouj, Moncef: Rate Adaptation for Dynamic Adaptive Streaming over HTTP in Content Distribution Network. In: *Image Commun.* 27 (2012), April, num. 4, pp. 288–311. – ISSN 0923-5965

[Liu et. al 2007] Liu, Tao ; Wang, Yao ; Boyce, J.M. ; Wu, Zhenyu ; Yang, Hua: Subjective Quality Evaluation of Decoded Video in the Presence of Packet Losses. In: *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on* vol. 1, April 2007, pp. I–1125–I–1128. – ISSN 1520-6149

[Lohmar et. al 2011] Lohmar, Thorsten ; Einarsson, Torbjorn ; Frojdh, Per ; Gabin, Frederic ; Kampmann, Markus: Dynamic adaptive HTTP streaming of live content. In: *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, June 2011, pp. 1–8

[Ma et. al 2014] Ma, He ; Seo, Beomjoo ; Zimmermann, Roger: Dynamic Scheduling on Video Transcoding for MPEG DASH in the Cloud Environment. In: *Proceedings of the 5th ACM Multimedia Systems Conference*. New York, NY, USA : ACM, 2014 (MMSys '14), pp. 283–294. – URL http://doi.acm.org/10.1145/2557642.2557656. – ISBN 978-1-4503-2705-3

[Ma et. al 2011] Ma, Kevin J. ; Bartos, Radim ; Bhatia, Swapnil ; Nair, Raj: Mobile video delivery with HTTP. In: *Communications Magazine, IEEE* 49 (2011), April, num. 4, pp. 166–175. – ISSN 0163-6804

[Michalos et. al 2012] Michalos, M.G. ; Kessanidis, S.P. ; Nalmpantis, S.L.: Dynamic Adaptive Streaming over HTTP. In: *Jounral of Engineering Science and Technology Review* (2012)

[Mok et. al 2012] Mok, Ricky K. P. ; Luo, Xiapu ; Chan, Edmond W. W. ; Chang, Rocky K. C.: QDASH: a QoE-aware DASH system. In: *Proceedings of the 3rd Multimedia Systems Conference*. New York, NY, USA : ACM, 2012 (MMSys '12), pp. 11–22. – ISBN 978-1-4503-1131-1

[Muller et. al 2013] Muller, Christopher ; Lederer, Stefan ; Pocher, Jorg ; Timmerer, Christian: libdash - An Open Source Software Library for the MPEG-DASH Standard. In: *IEEE International Conference on Multimedia and Expo Workshops (ICMEW)*, July 2013

[Nguyen and Ostermann 2007] Nguyen, Dieu T. ; Ostermann, Joern: Congestion Control for Scalable Video Streaming Using the Scalability Extension of H.264/AVC. In: *Selected Topics in Signal Processing, IEEE Journal of* 1 (2007), num. 2, pp. 246–253. – ISSN 1932-4553

[Oelbaum et. al 2008] Oelbaum, Tobias ; Schwarz, Heiko ; Wien, Matias ; Wiegand, Thomas: Subjective performance evaluation of the SVC extension of H.264/AVC. In: *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, Oct 2008, pp. 2772–2775. – ISSN 1522-4880

[Ohm et. al 2012] Ohm, Jens ; Sullivan, Gary J. ; Schwarz, Heiko ; Tan, Thiow K. ; Wiegand, Thomas: Comparison of the Coding Efficiency of Video Coding Standards x2014;Including High Efficiency Video Coding (HEVC). In: *Circuits and Systems for Video Technology, IEEE Transactions on* 22 (2012), Dec, num. 12, pp. 1669–1684. – ISSN 1051-8215

[Pantos and May 2010] Pantos, Roger ; May, William: HTTP Live Streaming / Apple Inc. 2010. –

[Papadimitriou and Tsaoussidis 2007] Papadimitriou, Panagiotis ; Tsaoussidis, Vassilis: A Rate Control Scheme for Adaptive Video Streaming over the Internet. Glasgow, Scotland, June 2007, pp. 616–621. – URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.79.9512`

[Parmar and Thornburgh 2012] Parmar, Hardeep ; Thornburgh, Michael: Real-Time Messaging Protocol (RTMP) Specification / Adobe Systems Inc. 2012. –

[Peng et. al 2008] Peng, Wen-Hsiao ; Zao, John K. ; Huang, Hsueh-Ting ; Wang, Tse-Wei ; Huang, Lin-Shung: A Rate-distortion Optimization Model for SVC Inter-layer Encoding and Bitstream Extraction. In: *J. Vis. Comun. Image Represent.* 19 (2008), December, num. 8, pp. 543–557. – URL `http://dx.doi.org/10.1016/j.jvcir.2008.08.002`. – ISSN 1047-3203

[Pozueco et. al 2013] Pozueco, Laura ; Pañeda, Xabiel G. ; Garcia, Roberto ; Melendi, David ; Cabrero, Sergio: Adaptable system based on Scalable Video Coding for high-quality video service. In: *Computers & Electrical Engineering* 39 (2013), num. 3,

pp. 775–789. – Special issue on Image and Video Processing Special issue on Recent Trends in Communications and Signal Processing. – ISSN 0045-7906

[Pozueco et. al 2014] Pozueco, Laura ; Pañeda, Xabiel G. ; GarcÃa, Roberto ; Melendi, David ; Cabrero, Sergio ; Orueta, Gabriel D.: Adaptation engine for a streaming service based on MPEG-DASH. In: *Multimedia Tools and Applications* (2014), pp. 1–20. – URL `http://dx.doi.org/10.1007/s11042-014-2034-y`. – ISSN 1380-7501

[Prasad et. al 2003] Prasad, Ravi. ; Dovrolis, Constantine ; Murray, Margaret ; Claffy, Kimberly: Bandwidth estimation: metrics, measurement techniques, and tools. In: *IEEE Network* 17 (2003), Nov, num. 6, pp. 27–35

[Radvision 2013] Radvision: Maintaining A High Quality Experience Over Unmanaged Networks / Radvision. 2013. –

[Rejaie et. al 1999] Rejaie, Reza ; Handley, Mark ; Estrin, Deborah: RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet. In: *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* vol. 3, 1999, pp. 1337–1345

[RFC1157 1990] RFC1157: A Simple Network Management Protocol (SNMP) / IETF, Network Working Group. RFC Editor, 1990. –

[RFC2326 1998] RFC2326: Real Time Streaming Protocol (RTSP) / IETF, Network Working Group. RFC Editor, 1998. –

[RFC2616 1999] RFC2616: Hypertext Transfer Protocol, HTTP 1.1 / IETF, Network Working Group. RFC Editor, 1999. –

[RFC3550 2003] RFC3550: RTP: A Transport Protocol for Real-Time Applications / IETF, Network Working Group. 2003. –

[RFC4566 2006] RFC4566: SDP: Session Description Protocol / IETF, Network Working Group. 2006. –

[RFC768 1980] RFC768: User Datagram Protocol (UDP). RFC Editor, 1980. –

[RFC793 1981] RFC793: Transmission Control Protocol (TCP). RFC Editor, 1981. –

[Rodgers and Nicewander 1988] Rodgers, Joseph L. ; Nicewander, W. A.: Thirteen Ways to Look at the Correlation Coefficient. In: *The American Statistician* 42 (1988), num. 1, pp. 59–66

[Schierl et. al 2010] Schierl, Thomas ; Fuente, Yago Sanchez de la ; Globisch, Ralf ; Hellge, Cornelius ; Wiegand, Thomas: Priority-based Media Delivery using SVC with RTP and HTTP streaming. In: *Multimedia Tools and Applications* 55 (2010), September, pp. 227–246. – ISSN 1380-7501, 1573-7721

[Schwarz et. al 2007] Schwarz, Heiko ; Marpe, Detlev ; Wiegand, Thomas: Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 17 (2007), num. 9, pp. 1103–1120. – ISSN 1051-8215

[Seo et. al 2010] Seo, Kwang-deok ; Kim, Jin-soo ; Jung, Soon-heung ; Yoo, JeongJu: A Practical RTP Packetization Scheme for SVC Video Transport over IP Networks. In: *ETRI Journal* 32 (2010), num. 2, pp. 281–291

[Seshadrinathan and Bovik 2009] Seshadrinathan, Kalpana ; Bovik, Alan C.: *Motion-based perceptual quality assessment of video.* 2009

[Shah and Nawaz 2014] Shah, Syed M. ; Nawaz, Tabassam: StreaMER: Streaming Media Edge Router, A Centralized Multimedia Transcoding Gateway for Multi-bit-rate Networks. In: *Journal of Basic and Applied Scientific Research* 4 (2014), num. 2, pp. 126–134

[Shao et. al 2010] Shao, Beilu ; Renzi, Daniele ; Amon, Peter ; Xilouris, George ; Zotos, Nikolaos ; Battista, Stefano ; Kourtis, Anastasios ; Mattavelli, Marco: An adaptive system for real-time scalable video streaming with end-to-end QOS control. In: *Image Analysis for Multimedia Interactive Services (WIAMIS), 2010 11th International Workshop on*, 2010, pp. 1–4

[Shen et. al 2008] Shen, Bo ; Tan, Wai-tian ; Huve, Frederic: Dynamic Video Transcoding in Mobile Environments. In: *MultiMedia, IEEE* 15 (2008), Jan, num. 1, pp. 42–51. – ISSN 1070-986X

[Sisalem and Wolisz 1999] Sisalem, Dorgham ; Wolisz, Adam: Towards TCP-friendly adaptive multimedia applications based on RTP. In: *Computers and Communications, 1999. Proceedings. IEEE International Symposium on*, 1999, pp. 166–172

[Sodagar 2011] Sodagar, Iraj: The MPEG-DASH Standard for Multimedia Streaming Over the Internet. In: *IEEE MultiMedia* 18 (2011), num. 4, pp. 62–67. – ISSN 1070-986X

[Sohn et. al 2010] Sohn, Hosik ; Yoo, Hana ; De Neve, Wesley ; Kim, Cheon S. ; Ro, Yong-Man: Full-Reference Video Quality Metric for Fully Scalable and Mobile SVC Content. In: *Broadcasting, IEEE Transactions on* 56 (2010), Sept, num. 3, pp. 269–280. – ISSN 0018-9316

[Sullivan and Wiegand 2005] Sullivan, Gary J. ; Wiegand, Thomas: Video Compression - From Concepts to the H.264/AVC Standard. In: *Proceedings of the IEEE* 93 (2005), Jan, num. 1, pp. 18–31. – ISSN 0018-9219

[Telchemy 2008] Telchemy: *Understanding IP Video Quality Metrics.* 2008 (Application Note)

[Thang et. al 2012] Thang, T.C. ; Ho, Quang-Dung ; Kang, Jung W. ; Pham, A.T.: Adaptive streaming of audiovisual content using MPEG DASH. In: *IEEE Transactions on Consumer Electronics* 58 (2012), February, num. 1, pp. 78–85. – ISSN 0098-3063

[Timmerer et. al 2007] Timmerer, Christian ; Devillers, Sylvain ; Ransburg, Michael: ISO/IEC 21000-7:2007 Part 7: Digital Item Adaptation / International Standardization Organization. 2007. –

[Tionardi and Hartanto 2003] Tionardi, Laurensius ; Hartanto, Felix: The use of cumulative inter-frame jitter for adapting video transmission rate. In: *TENCON 2003. Conference on Convergent Technologies for Asia-Pacific Region* vol. 1, 2003, pp. 364–368 Vol.1

[Wang et. al 2008] Wang, Bing ; Kurose, Jim ; Shenoy, Prashant ; Towsley, Don: Multimedia streaming via TCP: An analytic performance study. In: *ACM Transactions on Multimedia Computing Communications and Applications* 4 (2008), May, num. 2, pp. 1–22. – ISSN 1551-6857

[Wanxiang and Zhenming 2001] Wanxiang, Cheng ; Zhenming, Lei: An modified RTP adaptive algorithm. In: *Info-tech and Info-net, 2001. Proceedings. ICII 2001 - Beijing. 2001 International Conferences on* vol. 2, 2001, pp. 33–38

[Wenger et. al 2006] Wenger, Stephan ; Wang, Ye-kui ; Hannuksela, Miska M.: RTP payload format for H.264/SVC scalable video coding. In: *Journal of Zhejiang University SCIENCE A* 7 (2006), May, num. 5, pp. 657–667. – ISSN 1009-3095

[Wenger et. al 2007] Wenger, Stephan ; Wang, Ye-Kui ; Schierl, Thomas: Transport and Signaling of SVC in IP Networks. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 17 (2007), num. 9, pp. 1164–1173. – ISSN 1051-8215

[Wiegand et. al 2009] Wiegand, Thomas ; Noblet, Ludovic ; Rovati, Fabrizio: Scalable Video Coding for IPTV Services. In: *Broadcasting, IEEE Transactions on* 55 (2009), June, num. 2, pp. 527–538. – ISSN 0018-9316

[Wiegand et. al 2003] Wiegand, Thomas ; Sullivan, Gary J. ; Bjontegaard, Gisle ; Luthra, Ajay: Overview of the H.264/AVC video coding standard. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 13 (2003), July, num. 7, pp. 560–576. – ISSN 1051-8215

[Wu et. al 2000] Wu, Dapeng ; Hou, Y.T. ; Zhu, Wenwu ; Lee, Hung-Ju ; Chiang, Tihao ; Zhang, Ya-Qin ; Chao, H.J.: On end-to-end architecture for transporting MPEG-4 video over the Internet. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 10 (2000), num. 6, pp. 923–941. – ISSN 1051-8215

[Wu et. al 2001] Wu, Dapeng ; Hou, Y.T. ; Zhu, Wenwu ; Zhang, Ya-Qin ; Peha, J.M.: Streaming video over the Internet: approaches and directions. In: *Circuits and Systems for Video Technology, IEEE Transactions on* 11 (2001), Mar, num. 3, pp. 282–300. – ISSN 1051-8215

[Xin et. al 2005] Xin, Jun ; Lin, Chia-Wen ; Sun, Ming-Ting: Digital Video Transcoding. In: *Proceedings of the IEEE* 93 (2005), num. 1, pp. 84–97. – ISSN 0018-9219

[Yang et. al 2012] Yang, Jun ; Sun, Ying ; Wu, Y. ; Sun, Shuli: Joint H. 264/scalable video coding-multiple input multiple output rate control for wireless video applications. In: *Image Processing, IET* 6 (2012), num. 1, pp. 43–52

[Zambielli 2009] Zambielli, Alex: ISS smooth streaming technical overview. In: *Microsoft Corporation* (2009)

[Zhou et. al 2012] Zhou, Chao ; Zhang, Xinggong ; Huo, Longshe ; Guo, Zongming: A control-theoretic approach to rate adaptation for dynamic HTTP streaming. In: *2012 IEEE Visual Communications and Image Processing (VCIP)*, 2012, pp. 1–6

[Zinner et. al 2010] Zinner, Thomas ; Hohlfeld, Oliver ; Abboud, Osama ; Hossfeld, Tobias: Impact of frame rate and resolution on objective QoE metrics. In: *Quality of Multimedia Experience (QoMEX), 2010 Second International Workshop on*, 2010, pp. 29–34

Parte VI

Apéndices

Apéndice A

Glosario de términos

ADTE	Adaption Decision Taking Engine
ATM	Asynchronous Transfer Mode
AU	Access Unit
AVC	Advanced Video Codec
BPMN	Business Process Model and Notation
CBR	Constant Bit Rate
CDN	Content Delivery Network
CGS	Coarse-Grain Scalability
CIF	Common Intermediate Format
CPU	Central Processing Unit
CSRC	Contribution Source
DASH	Dynamic Adaptive Streaming over HTTP
DCCP	Datagram Congestion Control Protocol
DCT	Discrete Cosine Transform
DID	Dependency ID
DSL	Digital Subscriber Line
FEC	Forward Error Correction
gBSD	generic Bitstream Syntax Descriptions
GoB	Group of Blocks
GoP	Group of Pictures
GQM	Goal Question Metric

HLS	HTTP Adaptive Live Streaming
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
IPTV	Internet Protocol Television
ISO	International Organization for Standardization
LAN	Local Area Network
MANE	Media Aware Network Element
MCTF	Motion Compensated Temporal Filtering
MGS	Medium Grain Scalability
MIMO	Multiple Input Multiple Output
MOS	Mean Opinion Score
MPD	Media Presentation Description
MPEG	Motion Pictures Experts Group
MPLS	MultiProtocol Label Switching
MSE	Mean Square Error
NAL	Network Abstraction Layer
NALU	Network Abstraction Layer Unit
NAT	Network Address Translation
NS	Network Simulator
P2P	Peer to Peer
PC	Personal Computer
PDA	Personal Digital Assistant
PSNR	Peak Signal-to-Noise Ratio
QCIF	Quarter Common Intermediate Format
QID	Quality ID
QoE	Quality of Experience
QoS	Quality of Service
RFC	Request For Comments
RR	Receiver Report

RTCP	Real Time Control Protocol
RTCP-APP	Real Time Control Protocol - Application packet
RTP	Real-time Transport Protocol
RTSP	Real Time Streaming Protocol
SDES	Session Description Protocol Security Descriptions
SDP	Session Description Protocol
SNR	Signal to Noise Ratio
SR	Sender Report
SSIM	Structural Similarity Index Metric
SSRC	Synchronization Source
SVC	Scalable Video Coding
TC	Traffic Control
TCP	Transmission Control Protocol
TID	Temporal ID
UDP	User Datagram Protocol
UEP	Unequal Error Protection
URL	Uniform Resource Locator
VCL	Video Coding Layer
VoIP	Voice over IP
WLAN	Wireless Local Area Network
XML	eXtensible Markup Language

Apéndice B

Arquitectura de los sistemas y configuración de los escenarios

Los sistemas de estimación y adaptación propuestos comprenden labores de diseño e implementación de los mismos en entornos reales. Es necesario, por tanto, resumir brevemente las herramientas empleadas, así como su integración en el sistema propuesto y la configuración del escenario de evaluación.

B.1. Sistema adaptativo basado en transcodificación

A continuación se describe la arquitectura interna de los módulos del sistema de estimación propuesto en el capítulo 10.

La figura B.1 muestra la arquitectura y los distintos subsistemas que conforman cada elemento. Un servicio básico de *streaming* únicamente estaría formado por el servidor y el cliente. Sin embargo, con el fin de dotar al sistema de capacidad de adaptación a las condiciones de red, se añaden dos nuevos sistemas: el sistema de estimación y el sistema de codificación. La conjunción de ambos sistemas permiten adaptar la tasa de transmisión de los contenidos al ancho de banda disponible, tal y como se ha explicado en la sección 10.2.

Las herramientas que se utilizaron para implementar la arquitectura propuesta se detallan a continuación. Algunas de ellas han sido convenientemente modificadas para que den soporte al sistema de estimación y adaptación.

- Servidor *streaming*: *Darwin Streaming Server 5.5.5*[1]. Es un servidor de código abierto que permite el envío de información multimedia empleando los estándares RTP y RTSP. Las modificaciones realizadas incluyen la gestión de los paquetes RTCP recibidos con la información de las métricas calculadas en el cliente. También se incluye el módulo de estimación a partir de la información recibida y un módulo encargado del establecimiento de la conexión con el codificador para transmitir el valor de la tasa de codificación estimada.

[1] http://dss.macosforge.org/

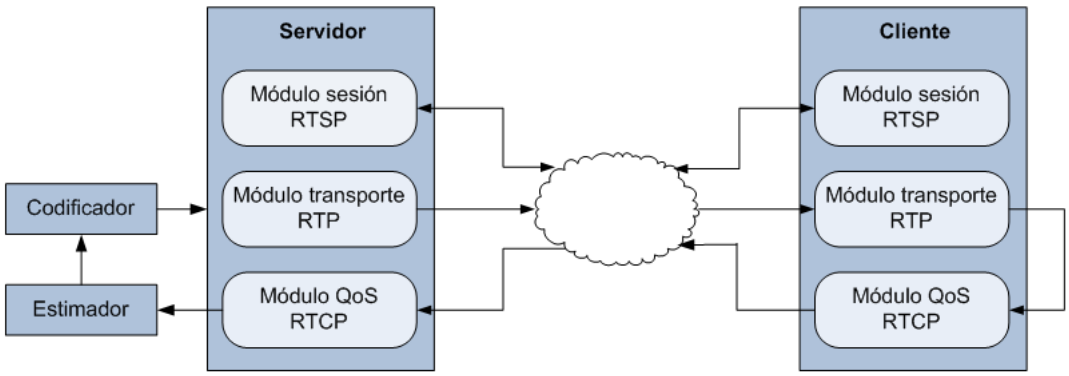


Figura B.1: Arquitectura del sistema basado en transcodificación

- Codificador adaptativo: *VLC Media Player 0.9.0*[2] + *Ffmpeg*[3]. VLC es un *software* de código libre que permite la reproducción y el envío de información multimedia. Incluye librerías para una amplia variedad de codificadores y decodificadores, proporcionados, en su mayoría, por librerías de *Ffmpeg*. En este caso, las librerías de *ffmpeg* son las encargadas de llevar a cabo la codificación adaptativa en formato MPEG-4.

- Cliente *streaming*: V*LC Media Player 0.9.0* + *Live555*[4]. El cliente de *streaming* VLC emplea el conjunto de librerías *live555* para gestionar las conexiones RTP, RTCP y RTSP. Las herramientas han sido modificadas para incluir el sistema de cálculo de las métricas de estimación y su posterior envío al servidor.

El criterio de selección de estas herramientas se basa principalmente en el carácter de código abierto y solución gratuita, con la funcionalidad necesaria para conseguir los objetivos y que además permiten realizar modificaciones para adaptarlas a los requisitos del sistema de adaptación propuesto.

En cuanto a la realización de los experimentos de evaluación del sistema implementado, se han empleado tres equipos. En el primer equipo instalaremos el servidor *straming* adaptativo. El emulador de red se ejecutará en una segunda máquina, mientras que la tercera máquina se reserva para la instalación del cliente *streaming* (Figura B.2). Las características técnicas de cada uno de los equipos se detallan en la Figura B.3. Para la configuración descrita, todos los equipos utilizan el sistema operativo *Ubuntu Linux* en su versión 8.04.

B.2. Sistema adaptativo basado en SVC

El sistema adaptativo basado en SVC (Capítulo 11) hereda la disposición de los módulos del sistema anterior basado en transcodificación, con la inclusión de un sistema de

[2]http://www.videolan.org/vlc/
[3]https://www.ffmpeg.org/
[4]http://www.live555.com/

Figura B.2: Despliegue físico

Equipo 1	
Procesador	Intel Core Duo CPU 3.40GHz
Memoria RAM	1 Gb

(a) Equipo 1

Equipo 2	
Procesador	Intel Pentium 4 CPU 2.80GHz
Memoria RAM	500 Mb

(b) Equipo 2

Equipo 3	
Procesador	Intel Pentium 4 CPU 2.80GHz
Memoria RAM	256 Mb

(c) Equipo 3

Figura B.3: Características técnicas de los equipos empleados en la evaluación del sistema basado en transcodificación

estimación y un sistema de adaptación en la arquitectura tradicional de un sistema *streaming* (Figura B.4). Las capas del vídeo escalable que se envían al cliente se controlan por medio de estos dos módulos extra, en función del ancho de banda disponible (ver Sección 11.2 con la descripción del algoritmo planteado).

Sin embargo, se simplifica ligeramente el desarrollo del sistema empleando plataformas reales. Para este caso, las herramientas modificadas únicamente comprenden el uso de dos librerías de código abierto:

- Servidor *streaming*: *live555*. Proporciona las librerías de código abierto para el manejo de sesiones *streaming* basadas en los estándares de RTP/RTCP/RTSP/SIP (Sección 6.3). En este caso, el desarrollo del servidor incluye la implementación de librerías que analicen los paquetes de *feedback* recibidos y estimen el ancho de banda disponible por medio de los algoritmos descritos en la Sección 11.2. Además, se implementa la funcionalidad necesaria para la inspección de las cabeceras de las unidades NAL y el envío de las capas correspondientes de vídeo al cliente.

- Cliente *streaming*: *openRTSP*[5]. Las librerías de *live555* se emplean en el desarrollo de diversas aplicaciones *streaming*. Entre ellas destaca el cliente *openRTSP*, que ha sido convenientemente modificado para que incluya el análisis de las cabeceras de los paquetes RTP para el cálculo de las métricas de estimación, que posteriormente serán empaquetadas y enviadas como *feedback* al servidor.

[5]http://www.live555.com/openRTSP/

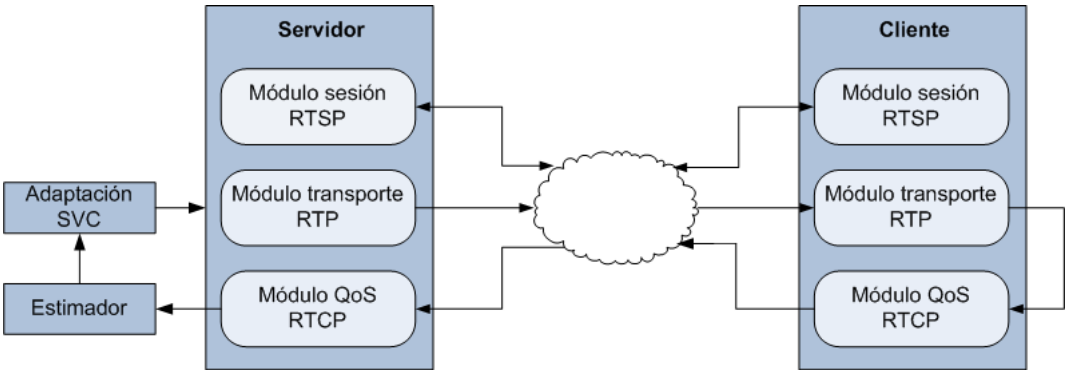


Figura B.4: Arquitectura del sistema basado en SVC

El despliegue físico para la evaluación del sistema se corresponde con el mostrado en la Figura B.2. Las características técnicas de cada uno de los equipos se describen a continuación: dos servidores Dell 850 (1xdual core, 2.4GHz, 1GB) para el servidor de vídeo y la herramienta de emulación NS3, y un servidor Dell R410 (2xquad core, 2.13GHz, 16GB) para los clientes.

Equipo 1	
Procesador	Intel Core Duo CPU 2.4 GHz
Memoria RAM	1 Gb

(a) Equipo 1

Equipo 2	
Procesador	Intel Core Duo CPU 2.4 GHz
Memoria RAM	1 Gb

(b) Equipo 2

Equipo 3	
Procesador	2xquad core 2.13 GHz
Memoria RAM	16 Gb

(c) Equipo 3

Figura B.5: Características técnicas de los equipos

B.3. Sistema adaptativo basado en DASH

La arquitectura del sistema adaptativo basado en DASH (capítulo 12) es sustancialmente diferente a los anteriores, debido a la naturaleza de la tecnología subyacente. En este caso, los módulos de estimación y adaptación se encuentran implementados en el cliente y los módulos del transporte de datos multimedia se corresponden con el modelo de petición-respuesta del protocolo HTTP (Figura B.6).

En este caso, el servidor no requiere el desarrollo o modificación de ningún tipo de *software*, de forma que emplearemos un servidor web HTTP convencional. Para el desarrollo del sistema adaptativo en el cliente, partiremos de la implementación oficial de referencia del estándar MPEG-DASH.

- Servidor *streaming*: Apache HTTP Server[6]. Es el servidor web HTTP de código abierto multiplataforma más empleado. Su configuración resulta sencilla y su popularidad se traduce en la disponibilidad de numerosos tutoriales y ayudas al soporte.

[6]http://httpd.apache.org/

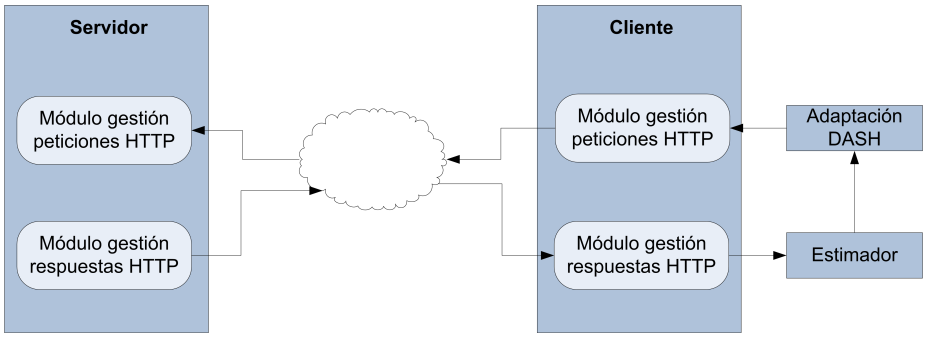


Figura B.6: Arquitectura del sistema basado en DASH

- Cliente *streaming*: libdash[7]. Es una librería de código abierto que responde a la implementación oficial de referencia del estándar MPEG-DASH. Proporciona la funcionalidad básica para la descarga y el manejo del fichero MPD y la descarga de los segmentos multimedia almacenados en el servidor HTTP. Para completar el cliente de streaming adaptativo, es necesaria la ampliación de las características básicas con los módulos de estimación y adaptación.

Las características técnicas de los equipos empleados en la fase de evaluación se corresponden con las descritas en las Figuras B.2 y B.5.

[7]https://github.com/bitmovin/libdash

Apéndice C

Emulación con NSE

Tradicionalmente, los métodos de simulación eran la opción adoptada para evaluar sistemas en entornos de red. Sin embargo, esto implica realizar una abstracción compleja y costosa, puesto que el sistema tiene que ser previamente modelado. Además, para elemenos reales como aplicaciones o usuarios, la precisión de las técnicas de simulación son dudosas.

La alternativa surge a partir del concepto de emulación, permitiendo mejorar la fiabilidad de los experimentos y evaluaciones. Podemos definir la emulación como un método de evaluación para servicios de comunicación que comprenden partes reales y partes modeladas, todas ellas ejecutándose en tiempo real.

La evalaución de los sistemas diseñados e implementados en la Parte IV de la documentación se realiza mediante técnicas de emulación, en las que las aplicaciones se ejecutan en un entorno real y el comportamiento de la red se simula mediante el software NS (Network Simulator).

NS es un simulador de eventos discretos que se emplea en campos como la enseñanza o la investigación con el objetivo de modelar redes IP. El éxito de NS-2 frente a otros simuladores de redes que existen en la actualidad (como *Opnet* u *Omnet++*) radica en el hecho de su naturaleza *open-source*, pudiendo disponer de un entorno gratuito y configurable. Es una herramienta muy potente, ya que permite modelar prácticamente todos los componentes de una red, desde el tráfico de aplicaciones tales como web, ftp, telnet, etc., con diferentes protocolos de transporte (TCP, UDP, *multicast*), con distintas políticas de enrutamiento y encolado en los elementos intermedios de la red y con diferentes tipos de enlaces (cableados, inalámbricos, satélite).

Desde la creación del proyecto hasta la fecha, las contribuciones en la implementación de la herramienta han dado lugar a dos versiones, NS-2[1] y NS-3[2]. NS-2 emplea dos lenguajes de programación: por un lado, utiliza C++ para implementar los protocolos. Por otro lado, para implementar las especificaciones de los escenarios que el usuario quiere evaluar, se emplea el lenguaje OTcl. Por su parte, NS-3 simplifica y unifica la programación de los escenarios, empleando el lenguaje de programación C++ tanto para implementar los protocolos como para implementar los escenarios diseñados por el usuario.

[1] http://www.isi.edu/nsnam/ns/
[2] http://www.nsnam.org/

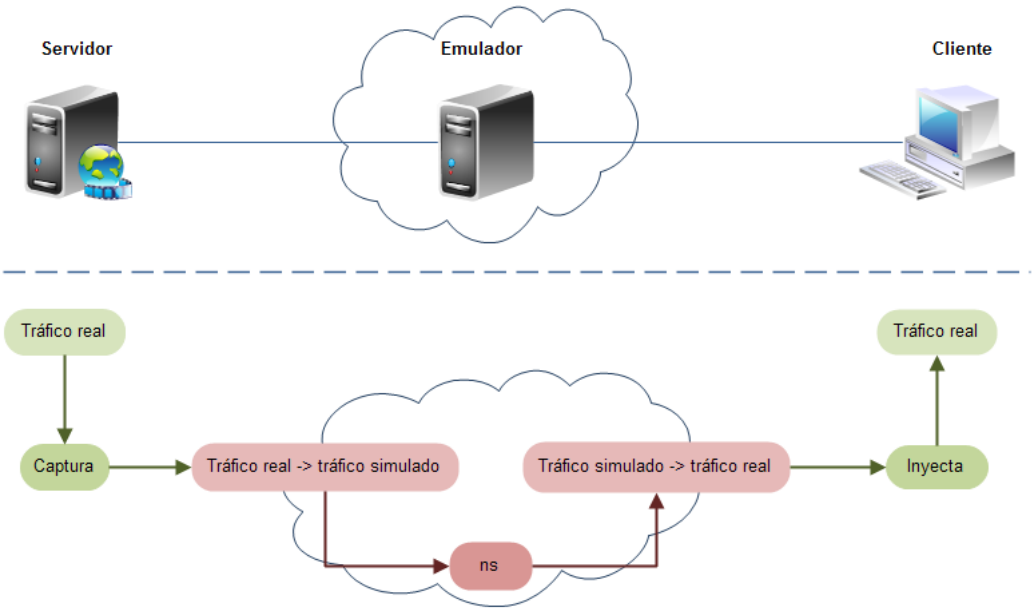


Figura C.1: Esquema del funcionamiento de NSE

Gracias a esta herramienta, en nuestros experimentos podremos crear una red emulada, por la que circulará el tráfico real proveniente de las aplicaciones de *streaming*. Por tanto, podremos estudiar el comportamiento del tráfico real frente a diferentes configuraciones de red, sin la necesidad de desplegar dichas redes físicamente. En nuestro caso resultará útil a la hora de variar el ancho de banda disponible en la línea para analizar el comportamiento del estimador, de manera que el tráfico real de las aplicaciones atraviese la red emulada, modificando así sus características de retardo, pérdidas, etc. en función de la configuración de la red virtual (Figura C.1). El resultado de la simulación o de la emulación se obtiene a través de trazas, que será necesario analizar a posteriori con herramientas como awk[3].

Los siguientes apartados muestran los modelos de variación de canal empleados para el análisis de cada uno de los sistemas de estimación y adaptación propuestos en los capítulos 10, 11 y 12.

C.1. Patrones de variación del canal para el sistema basado en transcodificación

Para realizar los experimentos de evaluación de las herramientas de estimación es necesario modificar en tiempo real el ancho de banda disponible en el enlace con el propósito de comprobar las labores de adaptación del algoritmo. En las pruebas utilizamos diferentes modelos que cubren diferentes situaciones que resultan interesantes a la hora de analizar el comportamiento del sistema. Dichos modelos se han implementado con la herramienta NS-2.

[3]http://www.gnu.org/software/gawk/manual/gawk.html

C.1.1. Modelo sierra ascendente

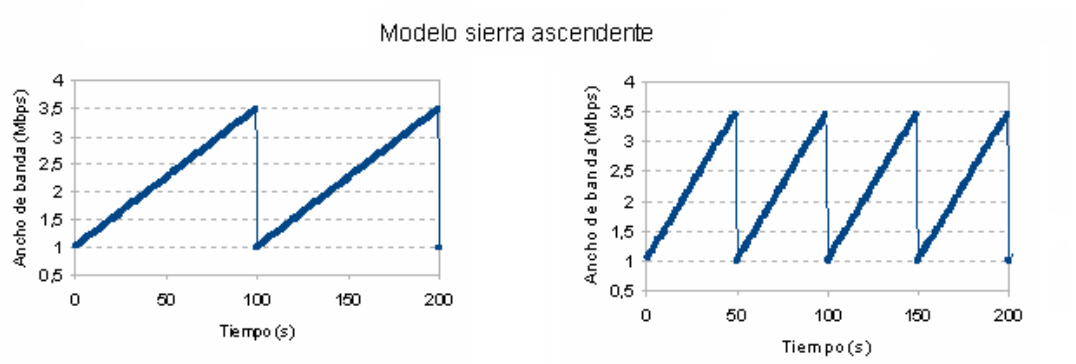


Figura C.2: Ejemplo del modelo sierra ascendente

Con este modelo podemos evaluar el comportamiento del estimador cuando la capacidad disponible en la red presenta una evolución creciente constante. Las tasas de variación se encuentran en el intervalo [1 Mbps, 3.5 Mbps]. El periodo de repetición el modelo tomó los valores de 50 y 100 segundos, de manera se obtienen dos escenarios análogos que presentan diferente pendiente ascendente.

C.1.2. Modelo sierra descendente

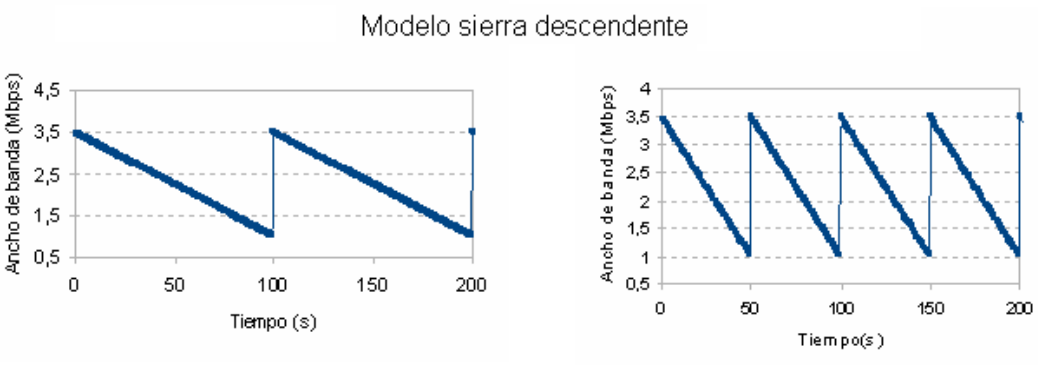


Figura C.3: Ejemplo del modelo sierra descendente

El objetivo de este modelo es el estudio del estimador cuando la capacidad del enlace disminuye. El ancho de banda disponible varía entre 1 Mbps y 5 Mbps. El periodo de repetición del modelo es de 50 segundos o 100 segundos.

C.1.3. Modelo triangular

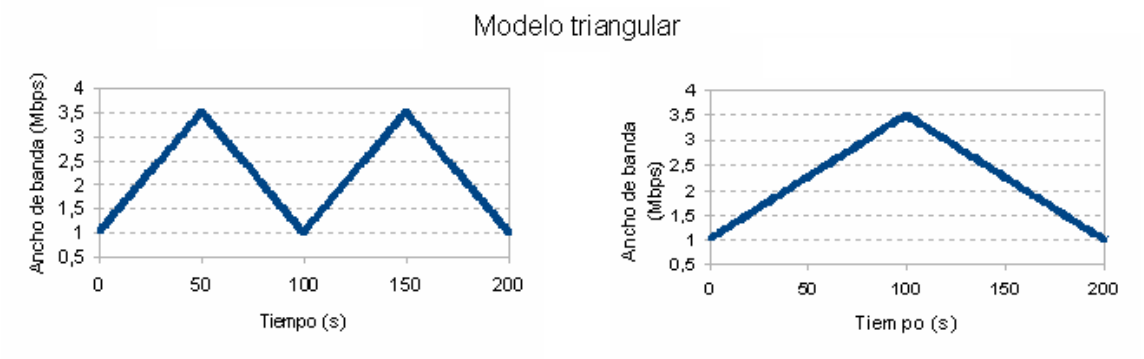


Figura C.4: Ejemplo del modelo triangular

En este modelo se combinan incrementos paulatinos en la capacidad del canal y descensos. Las tasas de variación se encuentran en el intervalo [1 Mbps, 3.5 Mbps]. El periodo de repetición el modelo tomó valores entre 50 y 100 segundos.

C.1.4. Modelo tendencia ascendente

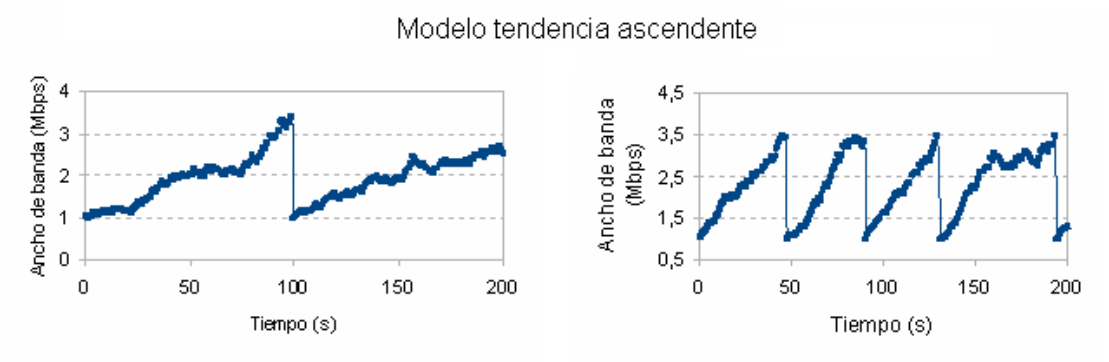


Figura C.5: Ejemplo del modelo de tendencia ascendente

En este caso el ancho de banda disponible mantiene una tendencia ascendente, tomando valores aleatorios. La tasa mínima permitida es de 1 Mbps y la tasa máxima es de 3.5 Mbps. El periodo de repetición del modelo, al igual que en los casos anteriores, oscila entre 50 y 100 segundos.

C.1.5. Modelo tendencia descendente

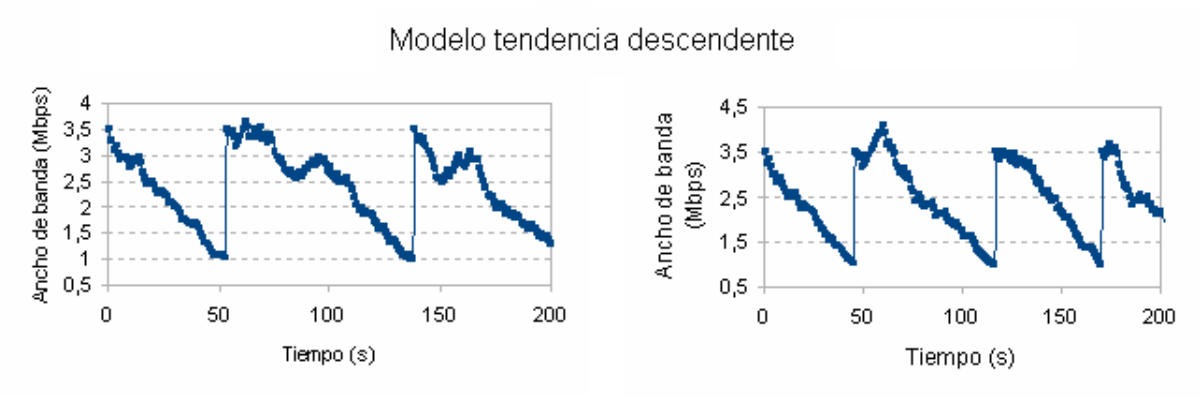


Figura C.6: Ejemplo del modelo tendencia descendente

Este modelo será análogo al anterior, con la diferencia de que el ancho de banda disponible comienza en un valor máximo y va disminuyendo.

C.1.6. Modelo escalón

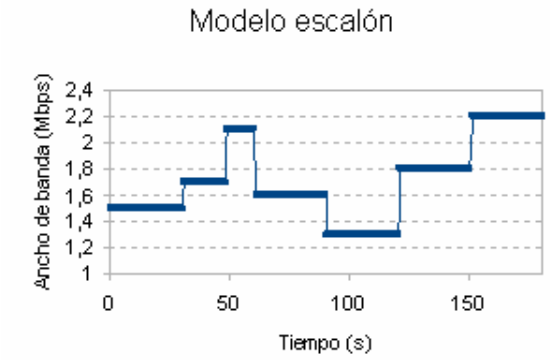


Figura C.7: Ejemplo del modelo escalón

Para este modelo la capacidad de la red se mantiene durante un periodo de tiempo, tras el cual, sufre un incremento o decremento instantáneo. La amplitud de esos cambios de capacidad del ancho de banda varía entre 800 Kbps y 100 Kbps.

C.2. Patrones de variación del canal para el sistema basado en SVC

En este caso, los modelos se han implementado con la herramienta NS-3. El carácter determinista de los patrones de variación del canal empleados facilitan las labores de repetición y duplicación de los escenarios por parte de la comunidad científica.

C.2.1. Modelo rampa ascendente

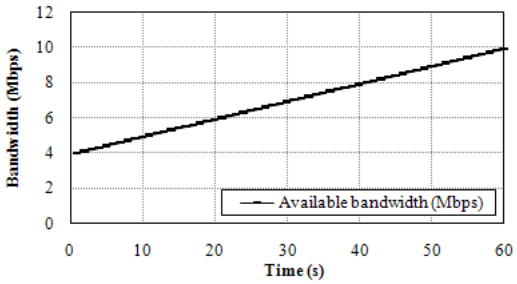


Figura C.8: Ejemplo del modelo rampa ascendente

La evaluación del comportamiento del estimador frente a incrementos en el ancho de banda disponible se realiza mediante el presente escenario. Las tasas de variación se encuentran en el intervalo [4 Mbps, 10 Mbps], siendo el periodo de repetición del modelo de 60 y 100 segundos.

C.2.2. Modelo rampa descendente

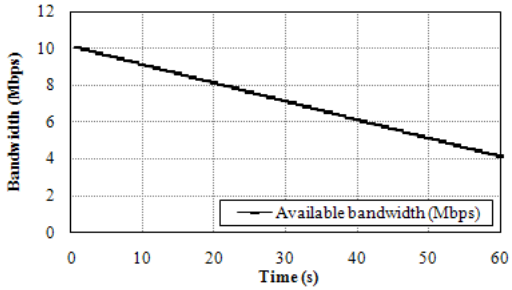


Figura C.9: Ejemplo del modelo rampa descendente

Escenario análogo al descrito anteriormente, pero con decrementos en el ancho de bada disponible.

C.2.3. Modelo escalón

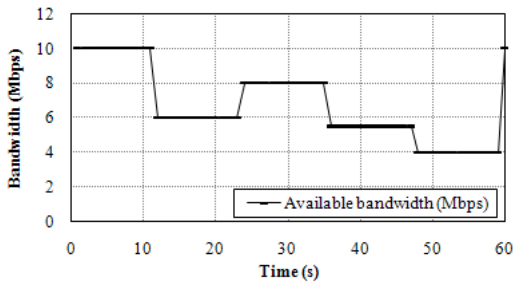


Figura C.10: Ejemplo del modelo escalón

Los incrementos o decrementos bruscos en la capacidad de la red se evalúan mediante escenarios de tipo escalón, como el presentado en la Figura. La amplitud de esos cambios de capacidad varía entre 1.5 Mbps y 4 Mbps.

C.2.4. Modelo escalón ascendente

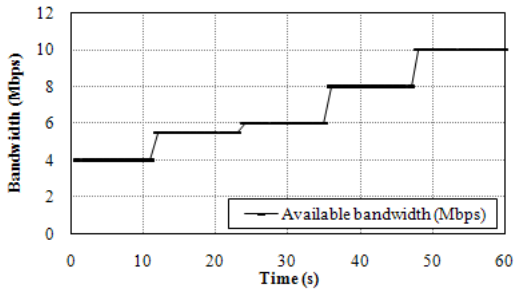


Figura C.11: Ejemplo del modelo escalón ascendente

En este modelo la capacidad de la red se mantiene durante un periodo de tiempo, tras el cual, sufre un incremento instantáneo. Permite evaluar la estabilidad del estimador durante periodos en los que no hay cambios en la capacidad de la red. La amplitud de esos cambios de capacidad podía variar entre 500 Kbps y 2 Mbps.

C.2.5. Modelo escalón descendente

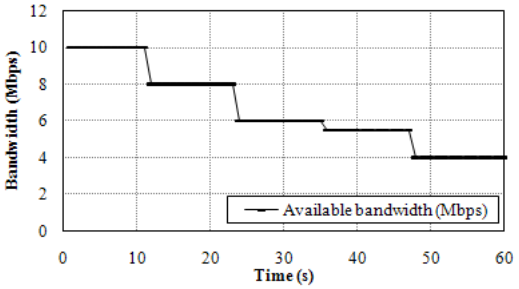


Figura C.12: Ejemplo del modelo escalón descendente

Escenario descendente análogo al anterior.

C.2.6. Modelo triangular ascendente

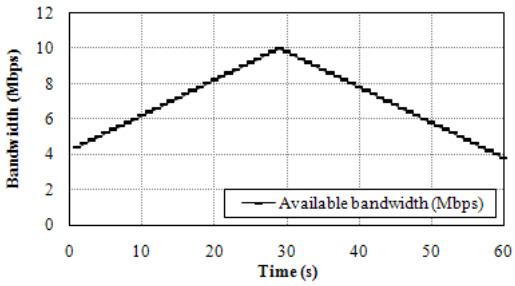


Figura C.13: Ejemplo del modelo triangular

En este modelo se combinan incrementos constantes en la capacidad del canal y decrementos. Permite evaluar la respuesta del estimador ante cambios en la tendencia del ancho de banda disponible. Las tasas de variación se encuentran en el intervalo [4 Mbps, 10 Mbps]. El periodo de repetición el modelo tomó valores de 60 segundos.

C.2.7. Modelo triangular descendente

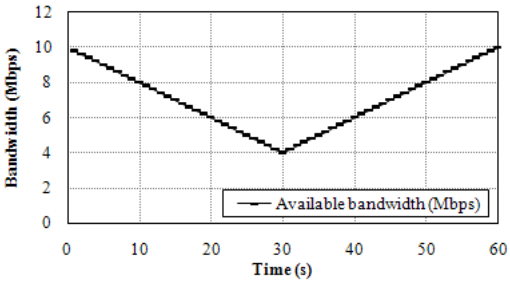


Figura C.14: Ejemplo del modelo triangular

En este modelo se combinan decrementos constantes en la capacidad del canal e incrementos. Las tasas de variación se encuentran en el intervalo [4 Mbps, 10 Mbps]. El periodo de repetición el modelo tomó valores de 60 segundos.

C.3. Patrones de variación del canal para el sistema basado en DASH

Siguiendo con las propuestas de variación del canal anteriores, se han implementado los siguientes modelos en NS-3 para evaluar el sistema adaptativo basado en DASH.

C.3.1. Modelo rampa ascendente

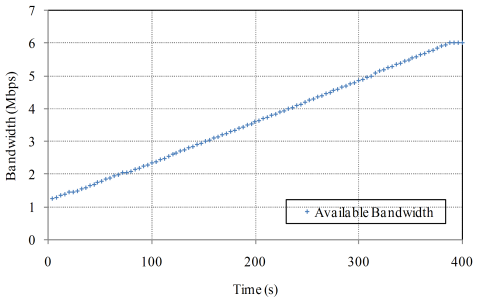


Figura C.15: Ejemplo del modelo rampa ascendente

El modelo propuesto permite evaluar el comportamiento del estimador ante incrementos continuos en el ancho de banda disponible. La tasa de variación del canal se sitúan en el intervalo [1 Mbps, 6 Mbps], siendo la duración completa del experimento de 400 segundos.

C.3.2. Modelo rampa descendente

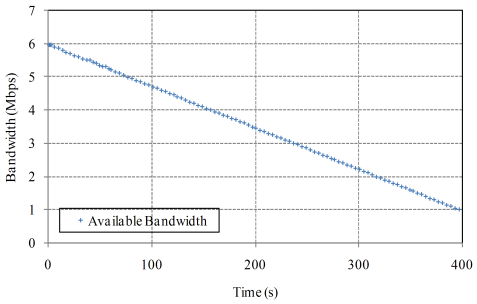


Figura C.16: Ejemplo del modelo rampa descendente

El escenario complementario al anterior presenta variaciones comprendidas entre 1 Mbps y 6 Mbps.

C.3.3. Modelo escalón

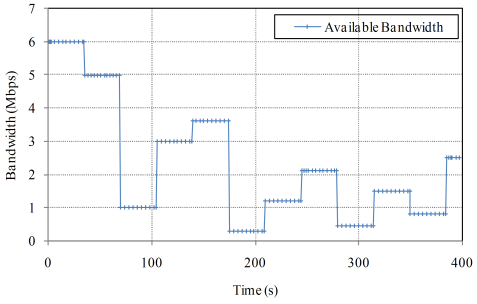


Figura C.17: Ejemplo del modelo escalón

El modelo escalonado permite comprobar la precisión del algoritmo de estimación y adaptación frente a cambios bruscos en la capacidad del enlace. En este caso, los cambios se presentan en el intervalo [300 Kbps, 6 Mbps]. La frecuencia de repetición del modelo es de 400 segundos.

C.3.4. Modelo escalón ascendente

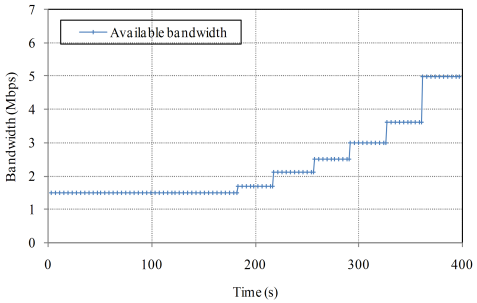


Figura C.18: Ejemplo del modelo escalón ascendente

La estabilidad del sistema frente a los incrementos en la tasa de transmisión es evaluado mediante modelos escalonados con tendencia ascendente. En este caso, la propuesta fija un intervalo de actuación comprendido entre 1.5 Mbps y 5 Mbps.

C.3.5. Modelo escalón descendente

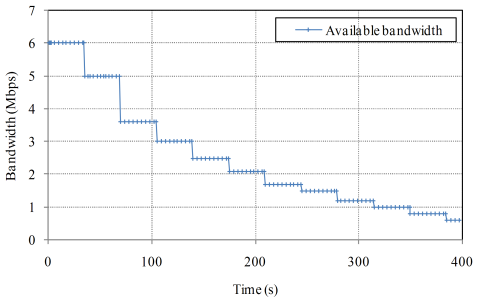


Figura C.19: Ejemplo del modelo escalón descendente

La disminución escalonada del ancho de banda disponible abarca el intervalo de [300 Kbps, 6 Mbps]. La duración completa del modelo es de 400 segundos.

Apéndice D

Realización de los experimentos

A la hora de realizar cada uno de los experimentos expuestos a lo largo de los capítulos 10, 11 y 12, es necesario tener en cuenta gran cantidad de variables, tales como la configuración del algoritmo, el modelo de variación de canal bajo estudio, el almacenamiento de los datos necesarios para calcular las métricas ... Todo este proceso se debe repetir de manera iterativa, con el fin de contar con numerosas muestras de cada experimento de forma que los resultados resulten coherentes.

A modo de ejemplo, la realización completa de un experimento podría incluir los siguientes pasos:

- Ejecutar la herramienta *tshark* en el servidor y cliente para capturar el tráfico que se generará durante la sesión *streaming.*
- Arrancar el servidor *streaming* que incluya las extensiones y modificaciones que den soporte al algoritmo de estimación y adaptación que se desea evaluar.
- Ejecutar el emulador de red NSE con la variación de ancho de banda bajo estudio. En este punto es importante añadir la ruta entre el cliente y el servidor (y viceversa) a través del equipo emulador, forzando así al tráfico a atravesar la red virtual.
- Arrancar el módulo de medición de actividad del sistema (SAR) en el cliente y servidor.
- Ejecutar el cliente, iniciando así una sesión *streaming.*
- Esperar a que finalice la sesión de *streaming.*
- Copiar el vídeo recibido en el cliente y las estadísticas generadas por éste, referentes a los valores de las métricas calculadas cada segundo.
- Copiar las estadísticas generadas en el servidor referentes a los valores de la tasa de transmisión en cada instante.
- Copiar datos generados y capturados con las herramientas SAR *y tshark.*
- Copiar los datos de emulación referentes al ancho de banda impuesto en cada instante.

La ejecución y repetición de todos estos procesos de una manera «artesanal» es una tarea laboriosa. Es por eso que se han diseñado una serie de *scripts* que permiten automatizar el proceso de elaboración de cada experimento, facilitando así la tarea de realización de los numerosos experimentos de laboratorio. Además, para disponer de cierto orden a la hora de realizar, almacenar y procesar los experimentos, se establecen una serie de niveles jerárquicos.

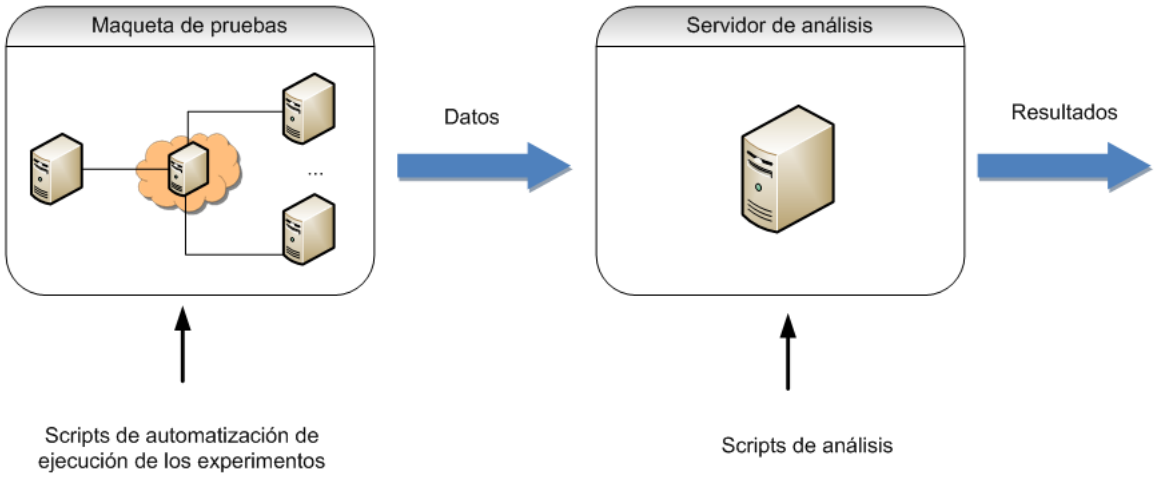


Figura D.1: Realización de los experimentos

El nivel más amplio que engloba a todos los demás es el nivel denominado "Experimento". Siguiendo la nomenclatura empleada que hemos visto a lo largo de las secciones del capítulo 10, ejemplos de experimentos son todos aquellos en los que el parámetro de variación es el peso que tienen las métricas dentro del algoritmo (0-100, 10-90 ...). En otros casos, otros ejemplos de experimentos difieren en el tipo de vídeo empleado para las pruebas. En definitiva, este nivel jerárquico abarca características globales que se desean evaluar.

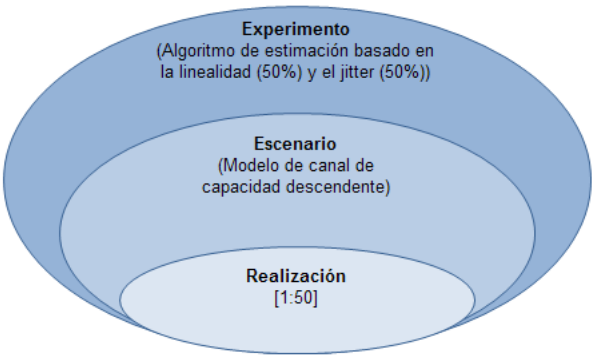


Figura D.2: Jerarquías en los experimentos

El siguiente nivel de diferenciación es el "Escenario". Aquí distinguimos entre los diferentes modelos de variación de canal empleados, esto es, modelos de tendencia ascendente, descendente, etc.

Por último, una vez elegido el tipo de experimento y el escenario, pasamos al nivel de realización, donde se lleva a cabo el ensayo de laboratorio propiamente dicho y donde se ejecutan de manera automática los pasos previamente descritos. Cada una de las realizaciones se repetirá un determinado número de veces, con el fin de obtener resultados estables.