# TESIS

Presentada el día 20 de Diciembre de 2016 en la

**Facultad de Ingeniería, Universidad de La República**

para obtener el título de

MAGISTER EN INFORMÁTICA

para

Pablo BERGER

PEDECIBA Informática
Instituto de Computación
FACULTAD DE INGENIERÍA
UNIVERSIDAD DE LA REPÚBLICA

Título de la tesis :

*Análisis de calidad para servicios de distribución de video adaptativo (HTTP Adaptive Streaming)*

Comité de examinadores:

| | | | |
|---|---|---|---|
| Dr. | Martín | VARELA | Revisor |
| Dra. | Adriana | MAROTTA | Examinadores |
| Dra. | Libertad | TANSINI | |
| Dr. | Pablo | RODRÍGUEZ-BOCCA | Director de tesis y académico |

## Agradecimientos

*En primer lugar quiero agradecer al Dr. Pablo Rodríguez-Bocca por su apoyo y estímulo permanente actuando como tutor en el estudio y el trabajo.*

*También quiero agradecer a los demás investigadores del área que han contribuido de una forma u otra con esta tarea.*

*Finalmente gracias a mi familia por el apoyo brindado en todo este tiempo.*

# Abstract

La infraestructura desarrollada de Internet, da hoy la posibilidad de utilizarla también para distribuir video, como una alternativa a los medios tradicionales de *Broadcasting*. Aunque la variabilidad que existe en los mecanismos de transporte de Internet no asegura un flujo de descarga constante. Al mismo tiempo la codificación de video tampoco generan un volumen constante de datos, dependiendo del tipo de contenido. Estas variabilidades afectan mucho en la disponibilidad de datos, frecuentemente se producen detenciones del desplegado, lo que afecta de forma notable la calidad percibida por el usuario.
En los sistemas de video se mantiene la dificultad para evaluar la experiencia del usuario como calidad percibida. En este caso de estudio se busca un cuantificador que evalúe en forma más simple y menos costosa la experiencia del usuario, con aproximación suficiente al valor obtenido en un ensayo *MOS* comparable. Se muestra para las tecnologías *HAS*, que tal cuantificador, se puede conseguir a partir de mediciones relativas a la disponibilidad y tipo de representación de los segmentos de video. También se presenta como, mediante ensayos es posible una evaluación del desempeño para el reproductor de video, lo que permite su mejora, con la finalidad de maximizar la calidad percibida.

Palabras clave: *MOS*, *HAS*, *Adaptive*, *Streaming*, *HTTP*

# Índice general

# Parte I

# INTRODUCCIÓN

# Capítulo 1

# Introducción

## 1.1 Motivación

En los últimos tiempos las aplicaciones basadas en video se están posicionando gradualmente como las más usadas en Internet y se prevé que serán dominantes en el consumo de recursos. Los protocolos que permiten el funcionamiento de Internet, aseguran una transferencia fiel o exacta de datos desde el servidor al cliente. Esto ha permitido el desarrollo de la *WEB* generando un acceso ubicuo a Internet, lo que ha reafirmado el uso de *HTTP/TCP/IP* y vice-versa [3].

El incremento en las capacidades de la infraestructura sobre la cual se desarrolla la Internet, da hoy la posibilidad de utilizarla también para distribuir servicios de video, como una alternativa a los medios tradicionales de *Broadcasting* de Televisión o de distribución por cable. Un ejemplo notable que muestra esta tendencia es *YouTube* [34], con más de mil millones de usuarios, luego de cumplir diez años desde su lanzamiento el 14 Febrero 2005. El número de horas que las personas miran videos usando este sistema al mes, crece un 50 % cada año. Actualmente 300 horas de contenido suben a esta plataforma cada minuto. La mitad de las reproducciones ocurren en dispositivos móviles y el 82 % los usuarios están entre 18 y 29 años. El crecimiento continuo de este tipo de plataformas ocupa una fracción de tráfico cada vez más importante en el total de Internet. En reportes de *Cisco Networking Index* [34] se proyecta un consumo agregado de 80 % al 90 % del total de tráfico de Internet para 2017. Solo *Netflix*, uno de los servicios por subscripción que lideran en este rubro en Estado Unidos y Canadá, consumió 29.7 % del tráfico pico de descarga para 2011 y 32.25 % en 2012.

En la actualidad no es raro que los dispositivos conectados a Internet puedan reproducir contenido multimedia distribuido desde la *WEB*, sin embargo un desplegado continuo y suave no está asegurado. Algunos problemas como tiempos de arranque prolongados o detenciones, a lo largo de la presentación, no son extraños. Esperas repetidas o prolongadas hacen bajar los indicadores de calidad percibida de los sistemas de video por Internet en comparación con los medios tradicionales [7].

Para el caso de señales de video, es necesario contar con la información a tiempo, man-

tener un flujo de datos constante para asegurar un desplegando continuo. La variabilidad que existente en los mecanismos de transporte de Internet no lo asegura, eso se traduce en tiempos variables para la finalización del proceso de descarga, como se muestra en la Sección 5.2, lo que afecta la calidad percibida por el usuario en el desplegado.

Para compensar estas deficiencias se implementan múltiples mecanismos, que van desde el uso de protocolos específicos, así como favorecer requerimientos mínimos aplicando políticas de QoS (calidad de servicio), en la red de transporte, con relativo éxito, a pequeña escala sin generalizarse en Internet. La infraestructura de Internet se comporta como una red *Best-effort* y por tanto no se adapta adecuadamente a aplicaciones con requerimientos real-time. El manejo de contenido de video necesita que las descargas de datos se puedan concretar con regularidad, y en tiempos limitados.

La oferta de contenido en formato de video se incrementa dada las nuevas facilidades para su generación y consumo por parte de los usuarios con sus nuevos dispositivos. La demanda se mantiene de manera sostenida y todas las estimaciones a futuro, presentan al video como el componente fundamental de tráfico en Internet. Por lo que el impulso [4], en el desarrollo de los servicios y los reproductores de video sigue en aumento. Al mismo tiempo los pronósticos apuntan que crecerá el consumo también en plataformas móviles, generando un tráfico agregado, con las particularidades asociadas a este tipo de redes. Para todos los casos, no es claro si es posible mantener indicadores *QoS* en forma constante para los dispositivos que consumen este tipo de servicios. Consecuentemente el impacto en la calidad percibida queda indeterminado.

La propia demanda en este tipo de servicios no es uniforme, la popularidad de los contenidos es muy dispar y cambiante con el tiempo, se presentan horas de uso intensivo o pico en donde estas situaciones adversas son notorias, al extremo de impedir el uso de algunos sitios o servicios en situaciones masivas. Las tecnologías han evolucionado con el fin de acotar estas anomalías de *HTTP/TCP/IP* para el transporte de información de video.

Actualmente las arquitecturas adaptativas, *HTTP Adaptive Streaming* [3], son un camino posible y dan una mejora para este tipo de servicios, ofreciendo una alternativa que permite mantener los indicadores de calidad percibida más uniformes por más tiempo. La mejora se consigue mediante una adaptación en el tipo de descargas, que realiza el reproductor de video. En estos casos los servicios ofrecen alternativas de codificación, para un mismo segmento de video, y el reproductor selecciona la secuencia más apropiada con el fin de obtener la experiencia de desplegado los más alta posible.

## 1.2 Objetivos

El objetivo de este trabajo es presentar las similitudes y contrastes de las tecnologías adaptativas actuales, para la distribución y reproducción de *Video Streaming* en *HTTP*. Mostrar como evolucionó en el tiempo, el estado del arte, partiendo de los conceptos básicos, presentando los protocolos y las arquitecturas principales.

Presentar métodos usados para estimar la calidad percibida en video. Desarrollar un indicador equivalente a *MOS* (Mean Opinion Score), y verificar que es suficientemente bueno para realizar una medición alternativa.

Analizar como evoluciona la calidad percibida en un enfoque general para uno de los reproductores más utilizados, y obtener conclusiones en relación con la facilidad de acceso que tiene al origen del contenido.

Entender como se manejan por parte de los reproductores de video las perturbaciones que externamente afectan al sistema. Utilizar el indicador propuesto como medida de eficiencia, en la evaluación de las diferentes alternativas existentes en los reproductores.

## 1.3 Contribución

En este trabajo se hace una investigación sobre temas referentes a las características básicas del *Video Streaming*, continuando con los protocolos usados actualmente y los diferentes ecosistemas técnicos o arquitecturas.

Al mismo tiempo se hace foco en los temas de calidad percibida de video y como los diferentes reproductores se comportan frente a a las condiciones *Best Efort* que da Internet para el acceso al contenido.

Se hace énfasis, en como las tecnologías adaptativas contribuyen a mejorar los aspectos de la distribución de *Video Streaming* en Internet y su impacto en la calidad percibida.

La evaluación de la calidad percibida para estas últimas tecnologías, se consigue en este trabajo con un estimador *MOS*. Mediante ensayos de laboratorio se verifica que este indicador es válido y proporciona valores comparables a los procesos tradicionales de cuantificación. Sobre el final se muestra como usar un indicador de este tipo para evaluar y mejorar las características de los reproductores en general.

## 1.4 Organización

Este documento se organiza en cuatro secciones principales e independientes.

En la primera parte se introducen conceptos básicos sobre la distribución de Video en Internet. Particularmente elementos relativos al dominio, como *CODECS* y la forma en que los protocolos evolucionaron desde *Progresive Download*, pasando por *Pseudo Streaming*, para llegar a los sistemas adaptativos. Se detallan estas últimas arquitecturas y como se usan en *VOD* y *LIVE*. Se introducen y se realiza un resumen planteando el problema que tienen en

común: *Microsoft's Smooth Streaming*, *Adobe's Dynamic HTTP Streaming*, *Apple HTTP Live Streaming* y *DASH*.

En la segunda parte se explica cuáles son los componentes fundamentales en formato de bloques, para un reproductor de video. También su modelo de funcionamiento general, haciendo foco en los mecanismos de decisión para mantener un desplegado continuo. También se muestran algunas opciones usadas actualmente para resolver este problema.

En la tercera parte se presentan temas relativos a la calidad percibida de video y también algunos métodos cuantitativos para su evaluación. Sobre el final se presenta una alternativa singular, para la cuantificación en las tecnologías adaptativas. Esta propuesta es una alternativa más simple y económica de evaluación que un ensayo *MOS* tradicional.

Finalmente en la última parte se detalla el experimento, que verifica en primer lugar, la validez del indicador similar a *MOS* usado. En segundo lugar se utiliza este método para la realización en otros ensayos que vinculan la calidad percibida con la capacidad de los reproductores para reaccionar a variabilidades externas.

# Capítulo 2

# Streaming de Video

## 2.1 Streaming

Se conoce con este nombre al proceso de estar constantemente recibiendo información desde un proveedor o servicio a través de una red de telecomunicaciones, haciendo uso de los mismos en forma inmediata o en diferido por parte del consumidor [4].

El nombre refiere principalmente al método de distribución y no al tipo de contenido en sí mismo. La televisión y la radio por Internet son algunos ejemplos clásicos de *Streaming* y han dado nombres particulares a los elementos que intervienen, relacionándolos con el tipo de contenido que se difunde.

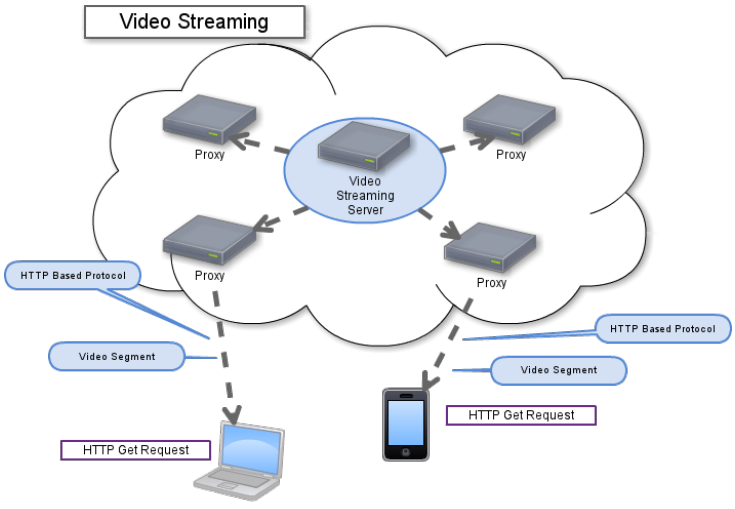


Figura 2.1: Descripción general de componentes para un sistema de *Video Streaming*.

La mayoría de los sistemas para distribuir contenido audiovisual en Internet presentan una estructura que se muestra en la Figura 2.1. Sin perder generalidad, en un sistema de *Video Strea-*

*ming*, se encuentran tres partes bien definidas como elementos componentes: un *Reproductor*, un *Origen* del contenido, y un mecanismo de transporte o distribución de datos que conecta a los primeros o *CDN*. Una *CDN* (Content Delivery Network) [9] [21] [29], es un conjunto de servidores, geográficamente distribuidos que absorben toda la carga de la distribución a los usuarios finales. En síntesis los componentes principales son:

- *Origen*: Toma las señales de entrada o contenido multi-media y codifica en forma digital en formato adecuado para la distribución.

- *CDN*: Atiende solicitudes y distribuye el contenido, consiste en servidores y elementos activos de red (infraestructura física de Internet).

- *Reproductor*: Permite al usuario buscar, localizar y lanzar la descarga de contenido. Una vez seleccionado el contenido, descarga los recursos relativos del mismo y ensambla un flujo continuo de video, de tal forma que pueda ser desplegado al usuario final.

En el *Video Streaming* el contenido puede estar previamente disponible o se puede generar en el momento de la distribución, se clasifica por este criterio en dos tipos con características particulares, *VOD* o *LIVE*.

En el streaming *VOD*, como acrónimo del inglés de *Video On Demand* (contenido a demanda), el contenido está disponible antes de que sea solicitado por un consumidor para su reproducción. Está disponible en su totalidad en forma de alguna representación almacenada en el origen, en donde permanece inalterado luego que se habilita el acceso a los consumidores. En este esquema, es posible preparar el contenido previamente a la distribución sin importar cuanto demore este proceso.

Los reproductores *VOD* consumen el contenido en la medida que llega del origen, y se espera inicialmente contar con datos suficientes para comenzar el desplegado. En los momentos que el flujo de datos no permite mantener la continuidad, se detiene el desplegado a la espera de la llegada de más contenido. El punto de detención entonces, se comporta como un nuevo punto de inicio, repitiéndose un comportamiento similar al arranque del *Stream*. Este comportamiento es característico de los reproductores *VOD*, donde no se descartan secciones del contenido por interrupciones del flujo de datos.

Por otro lado el *Streaming* de tipo *LIVE* como traducción literal del inglés *Live Streaming* (contenido en vivo), el video no está disponible y no se conoce el contenido de antemano, sino que es una señal que se genera en el momento de la distribución, su codificación no puede hacerse de antemano. Se refiere a tomar el contenido multimedia, distribuirlo por la red y presentarlo en tiempo real. Esta restricción temporal condiciona la preparación de la señal y agrega dificultades al momento de la codificación y distribución en tiempo real. El proceso involucra una fuente de señal, un *Encoder* (codificador) para su digitalización, un punto de publicación

donde se hace disponible el contenido multimedia a los usuarios finales, y una *CDN* (Content Delivery Network) para distribuir. Los usuarios finales acceden al punto de acceso para obtener el contenido en tiempo real, que pueden decodificar y desplegar.

En el desplegado, si hay alguna interrupción del flujo de datos es común omitir secciones del contenido, con la finalidad de mantener la característica de video en vivo. Se continúa en un punto de reproducción lo más próximo en el tiempo a la señal que se está generando, se asume que el contenido que no llegó, ya no es de interés y se descarta.

Los protocolos que permiten la administración y el envío de datos desde el origen al consumidor, se pueden clasificar en dos tipos. Si la transferencia la inicia el origen donde está el contenido, es un *Steaming* de tipo *Push*, si se inicia por el lado del consumidor se conoce como de tipo *Pull* [3]. En el Cuadro 2.1 se muestran que propiedades tiene cada uno.

| | Tipo *Push* | Tipo *Pull* |
|---|---|---|
| Tipo de Servidor | *Streaming Server, Windows Media, Adobe Flash, Apple QuickTime* | *Web server, LAMP, Microsoft* |
| Protocolos | *RTSP, RTMP, RTP, UDP* | *HTTP* |
| Formato *URL* | *rtsp:// ó rmtp://* | *http://* |
| Uso de ancho banda | Eficiente | Menos eficiente |
| *Buffer* en cliente | Si | Si |
| Monitoreo | Estándar | Particular de cada Arquitectura |
| *Multicast* | Si | No |
| *Content Caching* | No | Si |

Cuadro 2.1: Principales propiedades de protocolos tipo *Push* o tipo *Pull*.

## 2.2 CODEC

También es necesario la definición de formatos para el envío e interpretación de la información. El contenido se codifica en el origen y se decodifica en el reproductor, el componente de software que lo permite se llama *CODEC*. Es clave en el proceso, en el origen tiene como finalidad almacenar una secuencia de imágenes, provenientes de la señal, en un formato de datos propio de este componente. En el reproductor, decodifica los datos generados previamente y da como resultado un contenido desplegable. El formato específico de estos datos no es en principio relevante en el resto del sistema, queda confinado en la capa relativa al *CODEC*.

Con este componente de software se logra una representación comprimida de la señal original, por el descarte de información redundante que existe entre elementos de imágenes sucesivas (cuadros de video) en forma de agrupamientos o unidades auto contenidas. En *MP4* estas unidades de información pueden ser de-codificadas por separado unas de otras y son llamados *GOP* como *Group Of Pictures*, detalles en [18]. Cada uno de estos *GOP's* comienza con una imagen completa o *Key Frame* ("I" en la Fig. 2.2) a partir de la que se decodifican los demás cuadros del grupo. La compresión de la señal se logra en parte importante, debido a que las sucesivas imágenes, cuadros de la señal original son enviadas en forma diferencial respecto al *Key Frame*, (estas son llamadas "P" o "B" en la Fig. 2.2). La codificación también saca beneficios de redundancias espaciales y utiliza a favor proyecciones estadísticas.

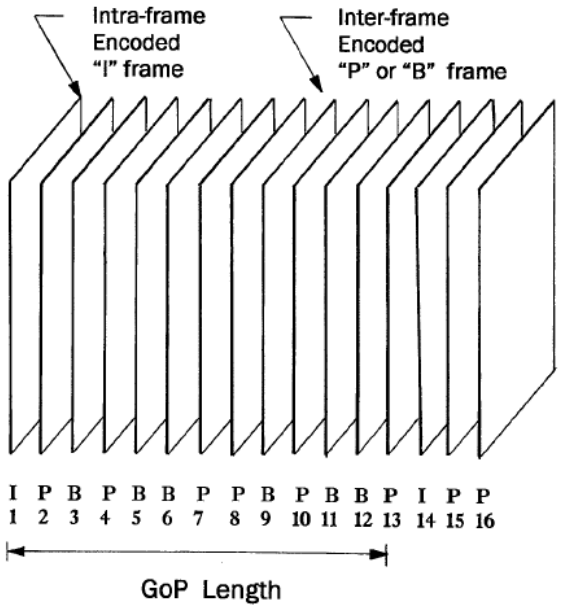


Figura 2.2: Group of Pictures.

La codificación es muy apropiada para almacenar o distribuir el contenido porque comprime mucho la información de la señal original, pero también agrega restricciones al sistema en su conjunto. Limita el comienzo de la reproducción de la señal original a un número de puntos en el tiempo, mucho menor que la cantidad de cuadros o *Frames* capturados, se debe comenzar la de-codificación en un *Key Frame*. Además se necesita un *GOP* completo para iniciar apropiadamente la de-codificación y que su resultado sea una secuencia de cuadros similares a la señal del contenido original, esto es porque en su estructura hay referencias cruzadas en el tiempo.

El *CODEC* se populariza según las propiedades de compresión que alcanza, y la fidelidad

que proporciona respecto a la señal original. Algunos de los más usados, son H264 y VP8 [1]. La representación de contenido codificado en H264 con audio AAC, tiene buenas propiedades de compresión y son procesados fácilmente por los reproductores que corren en el navegador [3].

En el funcionamiento del mecanismo de compresión se introducen errores respecto a la señal original. Este proceso es controlable en beneficio del ratio deseado o requerido para la compresión mediante el ajuste en los parámetros del *CODEC*, los más comunes son *Code Rate* (tasa de bits) y *Frame Rate* (tasa de cuadros). Luego de aplicar un *CODEC* al contenido, queda almacenable como datos con determinada *Calidad de Representación* o *Representación* de la señal original.

### 2.2.1 Code Rate (tasa de bits)

El *CODEC* pueden generar contenido codificado en *CBR (Constant Bit-Rate)* o en *VBR (Variable Bit-Rate)* [2]. En *CBR* se mantiene constante durante todo el contenido la tasa de codificación, el volumen de datos que se genera es proporcional al tiempo. Se recomienda el uso en los casos en donde el contenido es uniforme en movimiento o cuando los reproductores solo tienen esta opción. Si bien son mas fáciles de generar y consumir, se genera un volumen de datos constante, por momentos innecesario y en otros insuficiente para representar al contenido.

En *VBR* la tasa de bits con la que se codifica es variable, ajustándose hacia abajo o arriba según se necesite para codificar el contenido. El video con poca variación necesita generar menos información codificada. En estos codificadores se establece un límite superior a utilizar por el compresor. *VBR* toma más tiempo para codificar que el de tipo *CBR*, pero produce resultados más favorables en cuanto a la economía en datos y calidad de la representación. *VBR* es utilizado para el contenido de video de descarga progresiva o sistemas adaptativos, se considera de mejor calidad de compresión que *CBR*, pero se sugiere limitar la tasa de bits del máximo al 110 % de la media, para evitar fluctuaciones en la generación de datos [3].

[1] https://app.zencoder.com/docs/faq/codecs-and-formats

[2] http://help.encoding.com/knowledge-base/article/what-is-the-difference-between-cbr-and-vbr-encoding/

[3] http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/How-to-Produce-for-Adaptive-Streaming-81020.aspx

# Capítulo 3

# Protocolos de Streaming

En este capítulo se presentan opciones para el transporte de datos de contenido multimedia. Se muestran las principales propiedades de algunos protocolos utilizados actualmente, en relación con la forma de acceso al contenido y como se mantiene este vínculo. Se resalta en todos los casos el costo de mantener esta relación entre el origen y el destino, (sesiones). La presentación se extiende para los casos independientes de sesiones.

## 3.1 Dependientes de sesiones

Desde el punto de vista de los protocolos que permiten hacer el *Streaming de Video*, algunos mantienen un vínculo entre el reproductor y su servicio, una conexión de este tipo permite una re-alimentación del proceso general de desplegado [42]. En estos casos el reproductor envía información relativa al estatus del desplegado y recursos disponibles. Esta información es usada en el servicio para configurar dinámicamente parámetros en la codificación de la representación, que consume cada una de sus sesiones. Así se logra ajustar el sistema completo para dar una mejor visualización. Al mismo tiempo la realimentación con el servicio permite que el usuario envíe comandos como: desplazamientos, cambios de velocidad de reproducción o selección [4].

Un protocolo comúnmente usado con este fin es *RTSP* (Real Time Streaming Protocol [1], en la Fig. 3.1) que maneja sesiones y el *Playback*, a partir de pequeños paquetes *RTP* (Real-Time Transport Protocol) [41] que llevan el contenido. El protocolo de control llamado *RTCP* permite el mecanismo de realimentación hacia el servidor. En estos casos se asegura una distribución de contenido en tiempo real, como se requiere para el video, con un control estricto del servicio para cada sesión. [2] [3]

Estos protocolos diseñados específicamente para la distribución de contenido, tienen la

[1] https://www.jwplayer.com/blog/what-is-video-streaming/
[2] http://www.ietf.org/rfc/rfc2326.txt
[3] http://www.ietf.org/rfc/rfc2205.txt

desventaja que deben utilizar puertos, que generalmente son bloqueados por cortafuegos (*Firewalls*) y servidores de *Proxy* muy usados en la infraestructura de red hoy en día. Esto dificulta notoriamente la distribución del contenido multimedia por la red, donde la infraestructura de *CDN* o de *Proxy/Caching* se ha desarrollado naturalmente con la *WWW* en Internet.

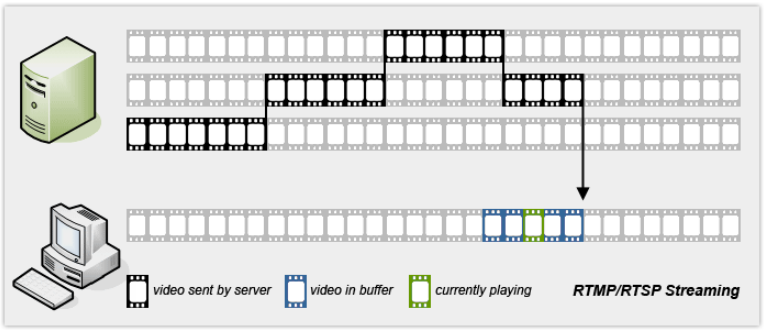


Figura 3.1: RTMP-STSP

### 3.1.1 IPTV

La *IPTV* es una opción de *Streaming* de video disponible en muchas geografías, opera por lo general con pequeños paquetes *RTP* para el transporte del flujo de datos [43]. En cuanto a requerimientos *Bandwidth* para SDTV (720×480 pixels) y HDTV (1920×1080 pixels) son 2–6, 6–16 Mbps Mbit/s respectivamente [4]. La difusión puede aprovechar el direccionamiento de tipo *IP Multicast*, lo que habilita el *Switching* entre canales de forma nativa. La codificación utilizada es de tipo variable *VBR*, que ahorra ancho de banda entre 40 % y 60 %.

*IPTV* tiene ajustes adaptativos frente a los cambios en las condiciones de la red, pero en este caso es el servidor el que ajusta el comportamiento de la demanda y cambia el tipo de la codificación a lo largo del tiempo mientras le entrega el contenido. El cambio de codificación en la representación del contenido se hace apropiadamente aprovechando las propiedades que da el *Framing*, para suavizar el desplegado.

En este caso el servicio mantiene una conexión con estado para cada una de las sesiones cliente, que el servicio usa para el ajuste de la codificación. Esto limita la escalabilidad del sistema a número máximo de reproductores que se pueden administrar.

### 3.1.2 RTMP

*RTMP* como *Real Time Messaging Protocol* usado por Flash Media Server y especificado en [19], [15] permite un *Streaming* adaptativo [4], como se muestra en la Fig. 3.1.

[4] https://www.jwplayer.com/blog/what-is-video-streaming/

Es posible el usar *RTMP* en túneles de *HTTP* como *RTMPT*, o también *RTMPE (Lightweight Encryption)*, *RTMPTE (Tunneling Lightweight Encryption)*, *RTMPS (Encrypted Over SSL)*, para permitir el cruce de *Firewalls*, es menos común.

Los protocolos antes descriptos, *RTMP*, *RTSP*, son usados para *VOD* y *LIVE Streaming*. En *LIVE* la situación no mejora, el proceso está más acotado en el tiempo, ya que el contenido está generándose y el servicio debe mantener una sesión para cada conexión adaptando la codificación. Estas restricciones temporales hacen aún más difícil, para el servicio, administrar las conexiones, por lo que estos sistemas tienen algún tipo de límite en la cantidad de conexiones que pueden admitir.

En ambos casos queda comprometida la escalabilidad del tipo de solución. Los reproductores mantienen una conexión con el servicio para dar retro-alimentación y al mismo tiempo también permiten que se envíen comandos para la navegación del contenido. Con una conexión establecida de este tipo, pequeños paquetes transportan el contenido, un mecanismo muy eficaz para el envío de datos, pero se dificulta el uso de infraestructura de Internet para el transporte y necesita ser encapsulado para pasar *Firewalls*.

## 3.2 Independientes de sesiones

En contraste con los protocolos anteriores, en donde se establecen conexiones permanentes a lo largo del desplegado del contenido, donde estos cierran un lazo de realimentación con el servicio, existe la alternativa de la descarga simple de datos. Se sigue usando *IP* como protocolo de red, pero se establecen sesiones más cortas sobre *TCP/HTTP* para lograr *HTTP Streaming*.

Si bien cada descarga establece una conexión entre el cliente y el servicio vía *HTTP*, este vínculo es menos permanente que en los casos anteriores y termina con el fin de cada descarga. No hay una sesión con el servidor para la administración de la transferencia del contenido. En algunos casos, la descarga es todo el video y se mantiene hasta que finaliza, aún en este caso con referencia al tratamiento del video podemos considerarlo sin un vínculo de realimentación hacia los servicios, ya que no permite que el cliente solicite cambios en la codificación una vez que el proceso se inicia. Si se requiere, debe tratarse entonces como si fuera una nueva descarga. En estos casos la descarga se hace en forma monolítica.

El uso de *HTTP Streaming* está fortalecido por la infraestructura de red asociada a Internet. Comúnmente los puertos usados en los servicios son 80 y 8080 ambos de TCP, estos no son bloqueados en los *Firewalls/Proxy* de la infraestructura de red, lo que facilita notoriamente la distribución del contenido [41].

Además este mecanismo se beneficia de las capacidades de almacenamiento en *Cache* provistas por las *CDNs*, lo que agiliza la distribución del contenido. En este caso son los reproductores los que administran el estado de la bajada del video, a través de listas de reproducción o algún esquema similar, mientras que los servicios, solamente presentan el contenido en forma

apropiada. Esto permite una escalabilidad mayor, por trasladar a cada cliente el control en las acciones de descarga, logrando el efecto contrario a consolidar la administración de las sesiones cliente.

Dentro de *HTTP Streaming* se puede dividir en dos casos bien definidos: a) Descargas monolíticas, como es el caso de *Progressive Download*, b) Descargas Segmentadas o *Pseudo-Streaming*.

Las descargas monolíticas solo pueden usarse como mecanismo para contenido *VOD* ; donde una lista de reproducción permite la elección inicial del contenido a desplegar. Como una extensión del caso a), se introduce una generalización de las listas de reproducción, con la segmentación del video.

Para el caso b), el contenido se fracciona en unidades, segmentos o fragmentos dependiendo de la implementación, que son accesibles vía servidores WEB a través de la URL a cada uno, de donde los clientes pueden obtenerlos. Estas unidades son relativamente pequeñas en comparación a todo el video, siempre es posible generarlos en forma estática para *VOD*, así como también en el momento para contenido tipo *LIVE*.

### 3.2.1 Descarga progresiva

Esta modalidad usa el primer tipo de descarga monolítica y está ampliamente difundida con extensiones particulares que mejoran su funcionamiento para *VOD*.

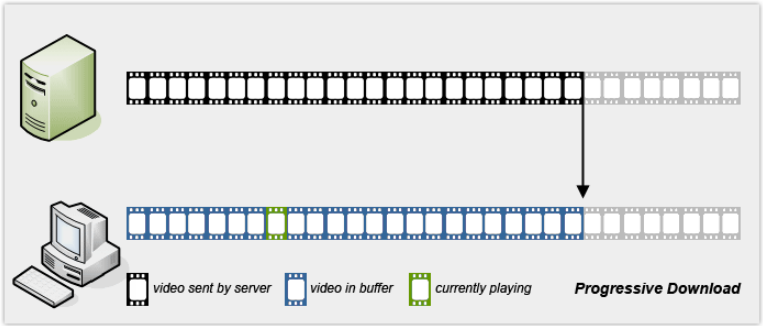


Figura 3.2: Progressive

Para cada video a demanda *VOD*, existe un archivo que lo representa y que se transfiere al reproductor cuando el usuario lo requiere. Esta transferencia no es diferente a otro tipo de descarga en cuanto al tratamiento del servidor. Sin embargo para datos de video no es necesario que se complete para comenzar a interpretarlo. Como se vio antes, el mecanismo de codificación-decodificación, permite regenerar, el contenido de un *GOP* en forma independiente de los demás. En la medida que existan suficientes de estos elementos auto-contenidos,

como para asegurar un flujo continuo, el sistema puede desplegar el contenido original en la codificación seleccionada sin interrupciones. De esta forma se acelera el comienzo de la reproducción, aún sin estar disponible la descarga total, se conoce con el nombre de *Progressive Download* [5], Fig. 3.2, frente al tradicional o *Legacy Download* que tiene la desventaja de enlentecer el comienzo del desplegado.

*Progressive Download* es muy simple y funciona sobre HTTP, se convirtió rápidamente en un sistema de descarga popular para videos cortos. Los casos conocidos son YouTube, Vimeo, MySpace, MSN y Soapbox. Uno de los inconvenientes de esta tecnología es su poca flexibilidad, no hay posibilidad de ajustar el tipo de descarga una vez que empieza. No es posible hacer un cambio de representación mientras se descarga y se observa el video. Antes de comenzar la descarga el usuario debe pre-seleccionar que tipo de representación prefiere. Si por alguna razón hay variaciones en la calidad de servicio o hay fluctuaciones del *Bandwidth* para las transferencia de la red, es muy factible que se experimenten puntos de detención o *Freeze* en la reproducción. Estas detenciones se mantienen hasta la llegada de nuevos elementos de la secuencia subsiguiente.

*Progressive Download* permite desplegar mientras continúa la descarga del contenido, por consiguiente los elementos de la secuencia llegan en forma ordenada y limitan la posibilidad de cambios rápidos en el punto de reproducción *Forward, Seek, Re-Play* para las primeras versiones.

En videos extensos aparecen problemas para mantener la descarga activa durante el desplegado del contenido, por lo que se usa frecuentemente para videos cortos del orden de 10 minutos de duración o con un tamaño aproximado a los 5MB. HTTP no es un protocolo orientado a conexión, por lo que mantener una descarga activa en el tiempo no es prioritario. En general no se mantienen las conexiones, terminan cuando la transferencia de datos finaliza. Por eso es dificultoso mantener una descarga por periodos prolongados de tiempo.

Además no es posible usar esta tecnología para distribución de video *LIVE*, porque no hay un contenido codificado previamente.

Algunas mejoras de *Progressive Download* se incorporaron en el transcurso del tiempo, fortaleciendo los servicios de este tipo, un ejemplo de ello es los 10 años de Youtube, con un esquema en el que se practicaron variaciones mínimas [42].

Una de las mejoras, está basada en el comportamiento de los propios usuarios. En general un video no se despliega en su totalidad, se descarta antes de finalizar. Como ejemplo, para un contenido de 10 minutos, es muy posible que el usuario, lo descarte a los 10 segundos de iniciado, desperdiciando ancho de banda, etc. La mejora *Bit Rate Throttling* aplicada, se implanta en el servidor de forma transparente para los clientes y quita prioridad a la descarga luego de una fase inicial, con la finalidad de liberar recursos en *Bandwidth* para aceptar nuevas cone-

[5] https://www.jwplayer.com/blog/what-is-video-streaming/

xiones. En un servicio sin *Bit Rate Throttling*, cuando se inicia una descarga, la transferencia se realiza a la mayor tasa disponible, hasta que se completa. Esto puede agotar los recursos disponibles en el servicio y limitar el inicio de una nueva. A pesar que algunos reproductores finalicen de forma anticipada, es decir, descartan el contenido antes de que finalice, el servicio sigue usando su máxima capacidad en la transferencia. Este mecanismo permite un uso más optimizado de la capacidades, y consigue aceptar nuevas descargas en forma simultánea. La entrega del contenido se hace a la mayor tasa disponible en los momentos iniciales (algunos segundos), esto da un *Fast-Start* o arranque rápido. Luego los servicios aplican reglas, según la tasa de codificación del contenido, asegurando un flujo de datos continuo, pero limitando la descarga muy por debajo de la máxima capacidad disponible.

### 3.2.2 Pseudo Streaming

Es el mecanismo dentro de *Progressive Download*, que permite el desplazamiento en el reproductor, a un sector del video que aún no ha bajado. Se llama a esta mejora *HTTP Pseudo-Streaming* [6], Fig. 3.3. Se consigue segmentando el contenido y tratando a cada segmento como una unidad para aplicar una descarga progresiva. [7] Así un punto de desplazamiento es considerado como un nuevo inicio, y solo se requiere un proceso de *Buffering* para continuar. El mecanismo de segmentación es indexado, y permite cambiar rápidamente el punto de reproducción, llevando el desplegado a la nueva posición, sin tener que descargar desde el origen todo el contenido.

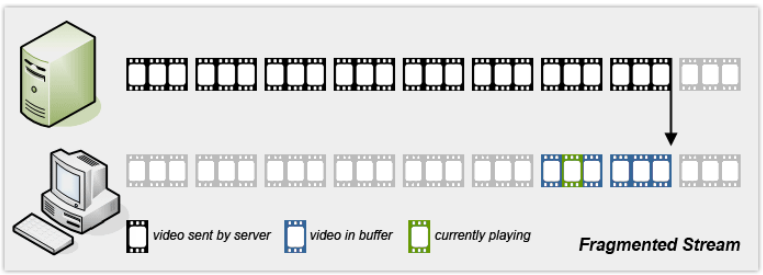


Figura 3.3: Pseudo Streaming

Este cambio aplicado al mecanismo de descarga, mantiene las ventajas del método original (permite el uso de infraestructura de Internet, no tiene problemas de pasaje por *Firewalls*) y mejora el desplegado en forma rápida de sectores del video que no estaban descargados.

Según el tipo de servidor, tipo de video se necesitan módulos para habilitar estas funcionalidades, por ejemplo para el caso de Apache y videos codificados en *MP4*, *H264 Streaming Module*.[8] Para Apache y formatos de video tipo FLV, *FLV Streaming Module*.[9] La funciona-

[6] https://www.jwplayer.com/blog/what-is-video-streaming/
[7] http://flash.flowplayer.org/plugins/streaming/pseudostreaming.html
[8] http://h264.code-shop.com/trac
[9] http://tperspective.blogspot.com.uy/2009/02/apache-flv-streaming-done-right.html

lidad de estos módulos es permitir el manejo de meta-data que tienen los videos. Se envía al comienzo del desplegado hacia el reproductor, una lista de los puntos de desplazamiento en tiempo de video, en donde existe un *Key Frame*, que puede asegurar una decodificación y desplegado eficaz, el salto en la descarga del archivo se hace con la directiva *Range* de *HTTP* estándar.

### 3.2.3 HTTP Adapvive Streaming

El éxito del *Streaming* de video basado en tecnologías de descarga progresiva, *Progressive Download* y *Pseudo-Streaming*, solo ha fortalecido la tendencia general al uso de *HTTP* como transporte eficaz, también para distribuir video. Esta evolución lleva a las tecnologías Adaptativas para *Streaming* sobre *HTTP* o simplemente *HAS* [10], Figura 3.4

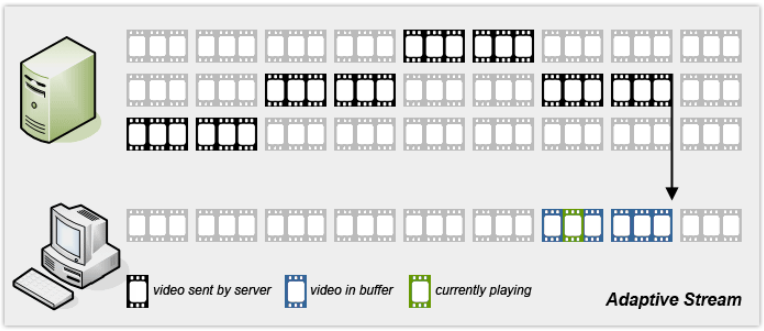


Figura 3.4: HTTP Adaptive Streaming

La flexibilidad que da el uso de la segmentación para las descargas permite aprovechar mejor las ventajas de Internet y el *Cache* existente en la infraestructura. La segmentación del contenido habilita a los consumidores a controlar mejor el proceso de descarga. Igual que para el caso progresivo el contenido segmentado permite desplazarse de un punto de reproducción a otro con facilidad sin necesidad de tener un vínculo con el servicio.

Cada consumidor, utilizando meta-data que envían los servicios o manifiesto adjunto publicado, obtiene información referente a como localizar los segmentos a concatenar para obtener una copia del contenido original.

Cuando se preparan los segmentos, se debe asegurar que cada uno comience con un *Key-Frame*, que permita la de-codificación en forma independiente de otros. Si esto ocurre, en forma ordenada y por número de segmento, se reconstruye el flujo o copia de la representación original.

[10] https://www.jwplayer.com/blog/what-is-video-streaming/

La segmentación funciona de igual manera para sistemas *VOD* como *LIVE*. Las diferencias aparecen en como se generan los contenidos y como se tratan luego por el mecanismo de reproducción. En lo que refiere estrictamente a la representación del contenido. En la práctica, si los segmentos se están difundiendo en el momento de su generación, es tratado como contenido *LIVE*, estos mismos, si son guardados, pasan a ser contenido de tipo *VOD* con el mismo manifiesto que se fue generando a medida que aparecen los segmentos.

Las tecnologías Adaptativas son una extensión de las descargas segmentadas. Un manifiesto especifica que tipos de *Representación* o también llamado tasa de codificación (Code-Rate) están disponibles para cada contenido requerido por el usuario.

Solo se requiere el agregado de una restricción adicional en la preparación del contenido:
"los segmentos de distintas representaciones de iguales tramos de contenido, deben tener como comienzo un *Key-Frame* en el mismo punto de tiempo de desplazamiento absoluto general". Como consecuencia de aplicar segmentación al contenido codificado y en simultáneo preparar representaciones más o menos fieles al mismo y respetando la restricción anterior, aparecen nuevas posibilidades, al momento de la descarga. Es posible ahora cambiar de *Representación* para el siguiente segmento a descargar y ensamblarlo al desplegar con los ya obtenidos sin que se presenten interrupciones, ya que cada segmento se puede decodificar en forma independiente. Ensamblar el video completo en el reproductor se logra con la ayuda de una extensión de lista de reproducción o manifiesto y se conoce como *Streaming* con capacidad de *Multibitrate*.

Este mecanismo permite flexibilidad para adaptar el tipo de descarga frente a cambios que son externos al reproductor de video. Estos cambios aparecen por inestabilidades en la red o en los servidores. Permite una flexibilidad mayor frente a las alternativas como descarga monolítica, *Progressive Download* y sus mejoras, ya que propone un mecanismo de ajuste gradual en contraste con ese tipo de soluciones todo o nada. Esto es favorable también para los servicios, en el sentido de no tener que llevar control sobre las descargas que realizan sus consumidores y permite servicios escalables.

HAS presenta un mecanismo, en donde es posible representar mismos intervalos de contenido, en forma de segmentos con distintos niveles de fidelidad a la señal original. Segmentos pequeños y menos estresantes para la descarga se ofrecen a costa de perder calidad en la representación de la señal original (errores de codificación). HAS deja la responsabilidad en la elección de la secuencia de segmentos a descargar a los propios clientes o reproductores, lo cual es una ventaja desde el punto de vista de la escalabilidad de la solución. No es necesario que el servicio lleve registro alguno de sus consumidores. Habilitando distintas alternativas para el consumo, en forma de segmentos de tamaños diferentes, *HAS* da la posibilidad al reproductor de video de encontrar una secuencia de segmentos que pueda consumir de acuerdo a las condiciones en las que se encuentre. Asumir de antemano, diferencias con respecto a la señal original, permite plantear a los sistemas que consumen video, un compromiso entre ofrecer una continuidad al desplegar frente a la exactitud de su representación. Esto no es posible en otros sistemas de distribución, a menos, de mantener una sesión con el servicio y un ajuste

personalizado en la codificación. Este es el problema que se estudia en este trabajo, sobre el cual se profundizará en los siguientes capítulos.

Algunas de las tecnologías que usan *HAS* actualmente son: *Microsoft Smooth Streaming*,[11] *HTTP Live Streaming*,[12] *HTTP Dynamic Streaming* [13] y se revisan con detalle en la Sección 4.1. Tienen muchos puntos similares en cuanto a la forma de operación o funcionamiento, pero difieren apenas en los formatos de sus protocolos. Un esfuerzo en este sentido está realizándose en *DASH*,[14] que promete estandarizar y dar compatibilidad cruzada.

Una ventaja implícita que tiene HAS por el uso de la segmentación, es que permite de igual manera y sin cambios en los formatos, la presentación de contenido *VOD* o *LIVE* solo tratándose de diferencias temporales en cuanto a la generación de contenido y al descarte de información que ya no está en tiempo real al momento del desplegado.

El esfuerzo en los servicios para generar múltiples codificaciones es importante, en especial para contenido en vivo o *LIVE*, pero es independiente de la cantidad de usuarios o consumidores. El acceso de los consumidores vía *CDNs* permite que la distribución pueda ser escalable en el sistema.

La implementación adecuada de los manifiestos, da información de localizaciones alternativas para los segmentos que se requieren, esto es muy conveniente y permite el uso de la *CDN* para la difusión del contenido generando un sistema global escalable y al mismo tiempo tolerante a fallas.

## 3.3 Resumen de protocolos

Comparativamente, si se observa el Cuadro 3.1, *HAS* es superior en varios aspectos, con excepción de la funcionalidad de desplazamiento dentro del contenido, la difusión tipo *Multicast* y un retardo extremo a extremo mas alto para *Live*.

[11] http://www.iis.net/downloads/microsoft/smooth-streaming
[12] https://developer.apple.com/streaming/
[13] http://www.adobe.com/products/hds-dynamic-streaming.html
[14] http://dashif.org/

| | RTSP/RTP | HTTP Progressive | HAS |
|---|---|---|---|
| Servidor | Propietario | Servidor Web | Servidor Web |
| Cliente | Difícil | Simple | Simple |
| Configuración | Difícil | Simple | Simple |
| Tipo | VOD, LIVE | VOD | VOD, LIVE |
| Buffer | Memoria | Temporal | Memoria |
| Desplazamiento | Total | Parcial | Parcial |
| Adaptación | En servidor | Ninguna | En reproductor |
| Bandwidth | Eficiente | Ineficiente | Suficiente |
| Tolerancia | Sin soporte | Sin soporte | Múltiple origen |
| DRM | Bueno | Malo | Muy bueno |
| Firewall | Difícil | Sin problemas | Sin problemas |
| Multicast | Bueno | No permite | No permite |
| Live delay end-to-end | 250mseg. | No permite | 20mseg |
| Uso típico | Real Time | Videos Cortos | Stream Dinámico |

Cuadro 3.1: Comparativo de funcionalidades

# Capítulo 4

# Video HTTP Adaptive Streaming

## 4.1 Arquitecturas HAS

Una alternativa viable para contribuir favorablemente en la difusión de *Video Streaming* es HAS, se han popularizado actualmente algunas propuestas que se consolidaron como opciones de arquitectura. Se muestran con detalle formatos y funcionamientos para el protocolo *HAS* en *Microsoft Smooth Streaming, Adobe Dynamic HTTP Streaming, Apple HTTP Live Streaming* y *Dynamic Adaptive Streaming HTTP*. Se describe cada una y se destaca que aspectos tienen en común.

En la Figura 4.1 se muestra un modelo general de *HAS*, según descripciones para la tecnología [17]. El contenido se fragmenta en el origen en una secuencia de *Segmentos* consecutivos y se publica su localización en forma de *Manifiesto*. Es importante en este esquema de arquitectura que los segmentos estén accesibles a los clientes vía *Get HTTP*, se describe en cada caso de que forma los servicios dan la funcionalidad. El rol de los servicios es atender la demanda de los segmentos pedidos por los reproductores clientes.

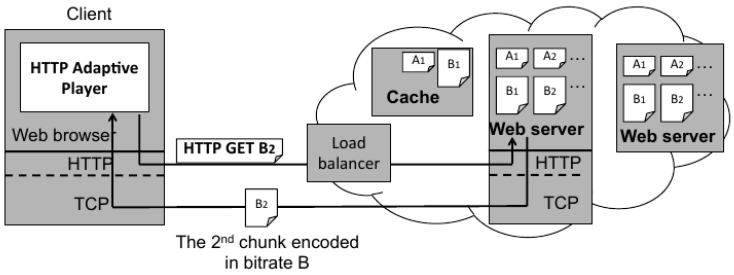


Figura 4.1: Reproductor o cliente de *HAS* descargando segmento de video requerido.

Los clientes de HAS descargan cada segmento, y recomponen la secuencia original. Cada uno de los segmentos está disponible en múltiples representaciones en el origen y el reproductor selecciona el que mejor se adapte a las condiciones de acceso.

La característica en el tipo de descarga de cada segmento, da al sistema la posibilidad de adaptación frente a la variabilidad en el acceso. Cada arquitectura resuelve como se lleva adelante la selección del próximo segmento de la secuencia. Repetidamente aparece el caso de estudio de este trabajo en cada una de las propuestas revisadas. Para este caso de estudio interesa, con que información y como hacen las *Heurísticas* esta selección.

En la Sección 5.2 de este documento se analizan con detalle las causas del problema general y se presenta de que manera se supera actualmente para cada caso.

## 4.2 Microsoft Smooth Streaming

La tecnología *Smooth Streaming* evolucionó desde las primeras versiones de adaptación, que comenzaron en los servicios de *Streaming NetShow 3.0* en 1998. En ese momento se habilitaba al reproductor de *Windows Media* para detectar el ancho de banda y reducir el deterioro en la presentación, manipulando la velocidad de fotogramas de la secuencia de video, en el peor caso el servidor sólo codificaba audio.

A esta modalidad de servicios le llamaron *True Streaming* o *Intelligent Streaming*, con la finalidad de garantizar de este modo un flujo continuo de datos para mantener la presentación del contenido aún con un rendimiento degradado de la red y para casos extremos solo la presentación de audio, lo que se consideraba la mejor opción para las velocidades de transferencia disponibles en el momento, comparables a conexiones de módem 28.8 kbps.

Entre 2000 y 2003, se introdujo el formato *Advanced Streaming Format, (ASF)*, que fue usado por mucho tiempo en este tipo de plataformas, lo que permitió al contenido tener varias presentaciones con bit-rates distintos e incluidas en el mismo archivo [8]. Estos formatos, protocolos e implementaciones propietarias daban al reproductor la habilidad para cambiar de flujo durante el desplegado, aunque no había forma de alinear las secuencias de video o de audio y se complicaba el cambio entre representaciones.

*Smooth Streaming* es una de las versiones comerciales de *HAS*, permite el cambio de *Stream* naturalmente y está construido sobre una extensión de la plataforma de alojamiento Web de *Microsoft Internet Information Service ( IIS ) 7.0*. Al igual que con otras técnicas de transmisión adaptativa, *IIS Smooth Streaming* se basa en el transporte de segmentos vía HTTP [42]. Las aplicaciones que consumen segmentos de video que ofrecen estos servicios, se han desarrollado utilizando la plataforma *Silverlight*, un subconjunto del *.NET Run-Time*, que se ejecuta enteramente dentro del navegador cliente. [1]

Para transportar el contenido previamente codificado vía *HTTP* se usa formato *ISO/IEC 14496-12 ISO, Base Media File Format* conocido como MP4. Las tecnologías previas en los

[1] https://www.microsoft.com/silverlight/

productos Microsoft no utilizaban MP4 sino ASF. MP4 es un contenedor más liviano para el procesamiento requerido dentro de los reproductores, necesita menos uso de CPU en comparación con ASF. La plataforma de desarrollo de Microsoft .NET tiene primitivas nativas que facilitan además su utilización.

MP4 fue desarrollado para transportar videos codificados en H264 y tiene incorporado soporte que permite la segmentación de la carga útil. En esta tecnología se llama fragmento a los segmentos de contenido, como *Fragmented MP4* (fMP4). La elección de MP4 por parte de Microsoft para mantener la representación del contenido, en los servicios se fundamenta en temas administrativos. Se argumenta que la segmentación o fragmentado, genera una gran cantidad de archivos pequeños que según explica Microsoft limita la escalabilidad y al mismo tiempo complica la administración.

El mismo tipo de contenedor MP4, es usado como formato para enviar al reproductor cliente los segmentos de contenido. Cada fragmento representa una secuencia de video de 2 a 4 segundos de duración, que puede ser recuperado del archivo, por su marca de tiempo índice equivalente y enviado en forma directa, sin otro procesamiento que el acceso indexado.

### 4.2.1 Componentes del servidor

Microsoft presenta una solución sofisticada para administrar el almacenamiento del contenido en los servicios, el tipo de contenedor MP4 permite un manejo indexado del contenido. Esta mejora en la manipulación del contenido, da escalabilidad en el manejo de cantidad de archivos involucrados. Mantener un mecanismo complejo en el servicio, con el fin de disminuir la cantidad de archivos a manipular, es un aporte de esta plataforma. Se argumenta, que se reduce mucho la sobrecarga en los servicios por la reducción de procesos de *Remuxing/Rewriting* involucrados en devolver el contenido almacenado, como se puede ver en [42].

No se habla de generación del segmento en el momento del envío, sino de una adaptación o conversión rápida de un formato de almacenamiento en disco o *Disk File Format* en Fig. 4.2 a un formato de transporte, también tipo MP4 reducido (subconjunto del primero) o *Wire Format* en fig 4.3.

Cada fragmento contiene una cantidad entera de *GOPs*, se forma así una unidad *MPEG-4 Movie Fragment* que es almacenado en MP4 para su acceso aleatorio en los sistemas de almacenamiento, conocido como *Disk File Format*. Ante un requerimiento *Get HTTP* del reproductor, el segmento se recupera y se envía en *Wire File Format*.

En el proceso de generación del contenido es fundamental que los *GOP's* "compartan" el mismo instante de tiempo para sus *Key-Frames*, esto es llamado alineación y permite conmutar segmentos durante el desplegado sin interrupciones.

Otro punto importante en la elección del tipo de contenedor está fundamentado en la posibilidad que tiene el formato de almacenar y enviar meta-data en forma distribuida para su con-

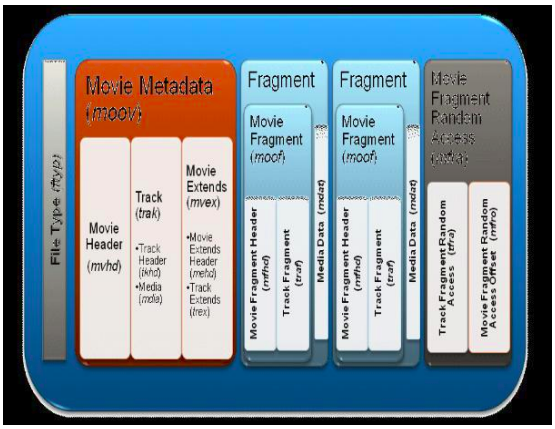


Figura 4.2: Disk File Format.

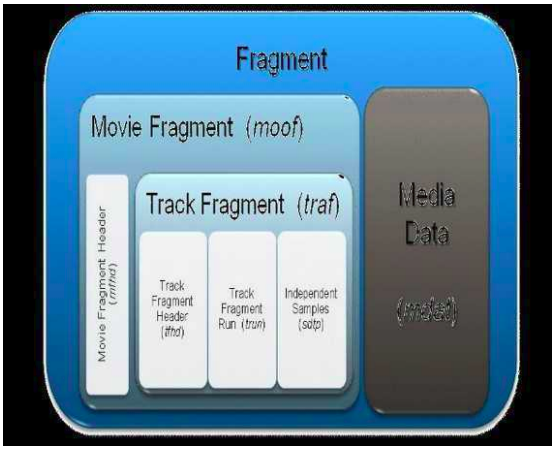


Figura 4.3: Wire Format

tenido. El aporte del formato en este sentido es dar la posibilidad sobre todo para los sistemas *LIVE*, en donde el contenido no es conocido de antemano, y se puede agregar esta información a medida que aparece. Hacerlo en forma distribuida en el contenido presenta menos sobrecarga inicial y un arranque rápido del *Stream*.

Por otro lado mantener la representación completa de un contenido en un archivo para cada opción de codificación o code rate, puede complicar en los mecanismos de replicación en la *CDN*.

### 4.2.2 Codificado y Publicación

La aplicación *Microsoft Expression Encoder* permite la creación de los archivos para *VOD* de forma transparente a los detalles o administrar un *Stream* para generar contenido *LIVE*. Esta aplicación genera los archivos en la representaciones adecuadas requeridas y los manifiestos

vinculantes que permiten la localización por parte de los clientes que consumen el contenido. [2].

En este proceso de preparación para difundir el contenido, el MP4 se almacena como un archivo de extensión *.ismv para el video y otro *.isma para solo audio [42]. Junto a estos se colocan un par de archivos de manifiestos, la extensión *.ism es para el uso del servidor y la otra *.ismc para que se procese en la consulta del reproductor. El manifiesto retornado al reproductor tiene formato *XML*, describe que opciones hay disponibles en el origen, para el contenido seleccionado, y es usado para localizarlos.

Los programas cliente utilizan *Smooth Streaming* y son generalmente interfaces integradas a medida, construidas sobre *Silverlight*. Inicialmente, para cualquier tipo de reproducción, el cliente solicita un manifiesto *XML* que contiene información sobre las representaciones disponibles para el contenido, en forma de una lista de segmentos de video. El reproductor puede entonces solicitar cualquier fragmento del servicio usando la URL asociada correspondiente.

La elección en la secuencia de segmentos a descargar no está previamente definida, es el reproductor que fija el criterio de descarga según heurística local en función de las capacidades de recursos y facilidad de acceso a los segmentos. La evaluación se hace en instantes predeterminados, aproximadamente cada 2 segundos o cuando hay espacio en el *Buffer*, dando como resultado cual es la representación apropiada para descargar, según pronóstico que asegure el mantenimiento de una cota con holgura, como se detalla en la Sección 6.2.

La secuencia de representaciones elegida tiende a mantener el *Stream* con el objetivo de maximizar la experiencia en el desplegado del contenido de video. La decisión es local para el próximo segmento a descargar, se pretende no hacer cambios o saltos abruptos, para obtener un comportamiento suave o *Smooth* evitando condiciones de parada *Freeze* o *Rebuffering*. Para que funcione esta tecnología adaptativa es necesario que los segmentos o fragmentos se puedan reproducir en forma independiente uno del otro. Para lograr esto se requiere como mínimo que cada segmento contenga un múltiplo entero de *GOPs*. Además los *Key-Frames* iniciales en cada uno, deben coincidir en el tiempo o estar "alineados" durante el proceso de codificación.

El reproductor los descarga utilizando para cada uno de ellos un esquema de *Progressive Download*, y se compone la secuencia de segmentos en un *Stream* para desplegar. La Figura 4.4 es un ejemplo de la sección del manifiesto en donde se definen que representaciones están disponibles, así como los desplazamientos o puntos de inicio de cada uno.

Con esta información el reproductor compone una URL de tipo, *http://video.foo.com /NBA.ism/QualityLevels(236000)/Fragments(video=41708333)* con la finalidad de obtener el segmento para la representación deseada *(Bitrate="236000")* en su desplazamiento desde el origen *"(41708333)"* expresado usualmente en unidades de tiempo cada 100 nanosegundos.

Para este tipo de pedido en el servicio se buscará el archivo MP4 correspondiente al conte-

[2]https://www.microsoft.com/expression/eng/

```
<QualityLevel Bitrate="370000" FourCC="H264" Width="384" Height="160" CodecPrivateData="0012456" />
<QualityLevel Bitrate="236000" FourCC="H264" Width="304" Height="128" CodecPrivateData="0012456" />
  <c n="0" d="41708333" />
  <c n="1" d="24190834" />
  <c n="2" d="40457083" />
```

Figura 4.4: Manifiesto de *Smooth Streaming*

nido, según el manifiesto invocado y con la información presente allí, se localiza el segmento por su *Track Fragment Random Access Index o TFRA*, se obtiene el fragmento *(MOOF + MDAT)* que es empaquetado y enviado como respuesta *HTTP*.

### 4.2.3 Cifrado

El cifrado se hace disponible a través de esquema para derechos digitales, *DRM PlayReady* propietario de Microsoft. Cuando se inicia el acceso al contenido cifrado, los clientes de *Silverlight* buscan al servidor de licencias del contenido en el servicio *PlayReady* para solicitar una licencia para su reproducción. *PlayReady* requiere que el contenido tenga una ruta segura entre el reproductor de video y el hardware de salida para contenido de alta definición. Debido a esto *Silverlight* no se puede usar en clientes de código abierto como Linux, a pesar de ser sólo un *Plug-in* para navegador. En resumen es importante destacar que esta tecnología necesita una configuración especial de componentes de software propietarios tanto para preparar el contenido así como para entregarlo según los requerimientos de sus reproductores.

## 4.3 Adobe Dynamic HTTP Streaming

*HTTP Dynamic Streaming* conocido como *HDS*, es una tecnología desarrollada por Adobe y Akamai para distribución de *Video Streaming* [12]. Los reproductores *HDS* se conectan a *Adobe's Flash Media Server*. Esta implementación proporciona un mecanismo de distribución flexible de video a demanda y en vivo en plataformas *Flash*. A partir de *Adobe Flash Player 10.1* se proporciona soporte para *HTTP Dynamic Streaming*, habilitando características como bitrate adaptativo, protección del contenido, aprovechamiento de dispositivos de *Cache* existentes y distribución basada en fragmentos con formato MP4 estándar *(F4F)*. *HTTP Dynamic Streaming* soporta tanto contenido en vivo como a demanda, ajustándose a la velocidad de conexión y capacidad de procesamiento del cliente, utilizando infraestructuras *HTTP* estándar que permiten atender la demanda a gran escala de contenido multimedia de alta calidad *(MPEG4 (H.264/AAC) o Flash video (FLV))*.

### 4.3.1 Componentes del servidor

HTTP Dynamic Streaming extiende el formato F4V utilizando un fragmento adicional en formato MP4 basado en estándares (ISO/IEC 14496-12:2008) con extensión .f4f. Este formato

permite que los pedidos HTTP obtengan y guarden en *Cache* porciones de contenido multimedia más pequeños. La especificación completa del formato F4F fue publicada por Adobe dentro de la especificación de F4V. Los detalles se pueden encontrar en www.adobe.com/devnet/flv. En la Figura 4.5 se muestra la secuencia de preparación, distribución, protección y consumo de video a demanda *VOD* y contenido en vivo usando *HTTP Dynamic Streaming*.

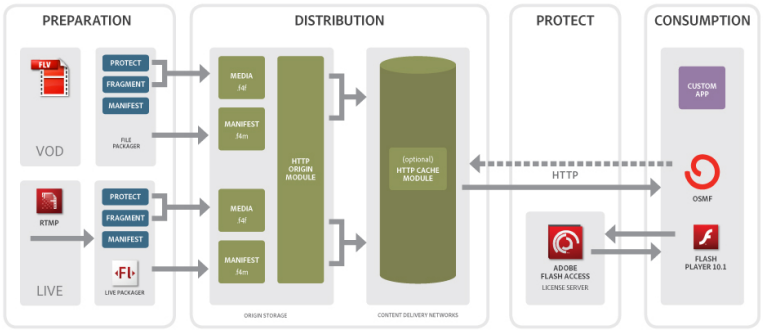


Figura 4.5: Funcionamiento

El empaquetado del contenido multimedia en el formato fragmentado es realizado por una de las siguientes herramientas:

- *File Packager*: es usado para preparar contenido pregrabado.
- *Live Packager*: es usado para preparar flujos *RTMP* en vivo.

Ambas herramientas realizan tres tareas:

- Generar archivos de fragmento MP4 (F4F).
- Generar un archivo de manifiesto basado en XML (F4M).
- Aplicar protección al contenido (opcional).

Para soportar la distribución del contenido en otras tasas de bits, el *File Packager* puede actualizar los archivos de manifiesto existentes con información adicional de tasas de bits. Notar que para soportar bitrate adaptativo, el contenido debe ser recodificado para asegurar alineamiento de *Key-Frames* entre los archivos de distintas tasas de bits [33]. El alineamiento es necesario para una transición correcta entre los distintos *Bit-Rates*.

Para contenidos en vivo, el proceso de empaquetado convierte cualquier flujo *RTMP* en el formato F4F. Los *CODECs* soportados para esto son H.264, AAC, VP6 y MP3. El *Live Packager* puede proteger el contenido *"on-the-fly"* y como salida genera múltiples segmentos F4F (conteniendo múltiples fragmentos). Análogamente al *File Packager*, también se crea un

archivo de manifiesto en tiempo real.

Los archivos de salida de los empaquetadores son colocados en un servidor *HTTP* clasificado como el origen. Este servidor es responsable de recibir pedidos de fragmentos por *HTTP* y retornar los segmentos correspondientes del contenido. Adobe proporciona un módulo para Apache para realizar esta tarea.[3]

### 4.3.2 Codificado y Publicación

Para asegurar una transición fluida entre las distintas representaciones del contenido, es necesario tomar algunas consideraciones al momento de codificarlo. Los *Key-Frames* iniciales de cada segmento deben estar alineados unos con otros, es decir representan el mismo instante absoluto en tiempo de Reproducción. Los *Key-Frames* son usados como guía para crear los segmentos, llamados fragmentos en esta implementación.

Flash solo puede cambiar de una representación a otra en un *Key-Frame*. Cuando están alineados entre todos los flujos segmentados, el cambio de una tasa de bits a otra se puede realizar sin que el cliente note un corte en la reproducción del contenido. Este alineamiento se consigue al forzar la configuración del *Encoder* para que utilice los mismos parámetros de *GOP (group of pictures)* para todos los flujos.

Los fragmentos del archivo completo son descargados y ensamblados por un cliente *Flash Player*. Mientras el flujo es reproducido, se monitorea el ancho de banda del cliente y se pueden llegar a cambiar los pedidos de fragmentos para un *Bit-Rate* más adecuado buscando mantener un flujo continuo.

Al comienzo de la reproducción el cliente descarga el archivo de manifiesto (F4M) que proporciona toda la información necesaria para reproducir el contenido, incluyendo el formato de los fragmentos, las tasas de bits disponibles, la dirección del servidor de licencias de *Flash Access* y meta información sobre el contenido. Las tareas de pedido y parseo del archivo de manifiesto, ensamblado de los fragmentos multimedia y monitoreo de la calidad de servicio de la red, requieren que se agregue lógica a los reproductores multimedia basados en Flash existentes, en forma de heurística que determine la secuencia de segmentos a obtener. Para simplificar esta implementación Adobe publicó el llamado *Open Source Media Framework (OSMF)* que proporciona soporte completo para *HTTP Dynamic Streaming*. [4] OSMF permite la decodificación del archivo de manifiesto, el cambio entre los distintos *Code Rates* y la administración de claves de protección del contenido. La Figura 4.6 muestra un ejemplo de la sección de manifiesto.

[3] www.adobe.com/go/httpdynamicstreaming
[4] http://sourceforge.net/adobe/osmf/home/Home/

**Multi-level Manifest**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns="http://ns.adobe.com/f4m/1.0" version="3.0">
<id>myVideo</id>
<baseURL>http://www.example.com/myvideo/</baseURL>
<streamType>live</streamType>
<media href="stream250.f4m" bitrate="250" />
<media href="stream500.f4m" bitrate="500" />
</manifest>
```

**Stream-level Manifest (stream250.f4m)**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns="http://ns.adobe.com/f4m/1.0" version="3.0">
<id>myStream1</id>
<baseURL>http://www.example.com/data/</baseURL>
<bootstrapInfo profile="named" id="boot1" fragmentDuration="4">
(BASE64 encoding of bootstrap information)
</bootstrapInfo>
<media url="stream250" bootstrapInfoId="boot1"/>
</manifest>
```

**Stream-level Manifest (stream500.f4m)**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns="http://ns.adobe.com/f4m/1.0" version="3.0">
<id>myStream1</id>
<baseURL>http://www.example.com/data/</baseURL>
<bootstrapInfo profile="named" id="boot1" fragmentDuration="4">
(BASE64 encoding of bootstrap information)
</bootstrapInfo>
<media url="stream500" bootstrapInfoId="boot1"/>
</manifest>
```

Figura 4.6: Manifiesto en dos niveles de *Dynamic HTTP Streaming*

### 4.3.3 Cifrado

Para contenido pre-grabado, el proceso de empaquetado crea un conjunto de archivos F4F a partir de un archivo previamente codificado en formato FLV o F4V. El proceso también genera un archivo de manifiesto (F4M) que describe el contenido y que potencialmente incluye meta data para manejar derechos digitales *DRM* como por ejemplo la dirección del servidor de licencias llamado *Flash Access*.

## 4.4 Apple HTTP Live Streaming

*HTTP Live Streaming* es la implementación HAS de Apple [24], conocida como *HLS*. Este mecanismo fue diseñado para poder distribuir audio/video de tipo *LIVE/VOD* desde un servidor, con la finalidad de hacerlos disponibles para reproducir en dispositivos móviles de Apple *iPhone, iPad, iPod touch* de forma nativa, así como en computadoras de escritorio con algunos reproductores de *HLS*. Inicialmente no era posible reproducir un *Video Streaming* en estos dispositivos sin usar *HLS*, lo que aceleró su utilización [3].

*HLS* soporta múltiples flujos alternativos codificados en representaciones distintas del contenido original, con la posibilidad para el reproductor cliente, de cambiar el flujo al que esté conectado. Estas decisiones son tomadas localmente, según disponibilidad en recursos locales y de cambios en las condiciones de acceso a los segmentos. El acceso a los segmentos está condicionado a las fluctuaciones de ancho de banda de la red, mal funcionamiento en los servicios o la propia capacidad para decodificar o desplegar localmente la representación del contenido.

### 4.4.1 Componentes del servidor

En una configuración típica, un codificador de hardware toma la entrada de audio/video y la codifica como *MPEG-4*, y genera como salida un flujo de transporte *MPEG-2*. Ese flujo es separado en una serie de archivos multimedia por el software de segmentación. Luego son colocados en un servidor Web. El software de segmentación a su vez genera y mantiene un índice o manifiesto que contiene una lista de los archivos multimedia generados. La *URL* del archivo índice se publica en el servidor *WEB*. El cliente descarga y lo lee para luego solicitar los segmentos multimedia en orden y presentarlos.

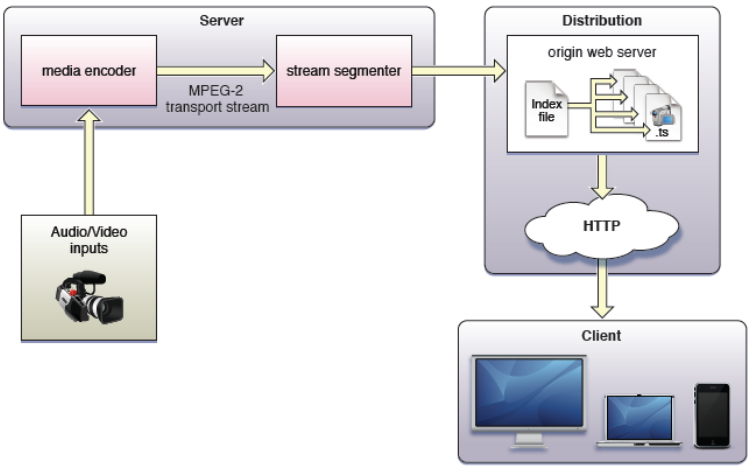


Figura 4.7: Funcionamiento

La entrada para el proceso puede ser en vivo o una fuente pre-grabada. Está típicamente codificada en un flujo de transporte *MPEG-2*. El flujo *MPEG-2* es separado en segmentos y guardado como una serie de archivos .ts. Los flujos exclusivamente de audio pueden ser series de archivos elementales de audio *MPEG* formateados en *AAC* con encabezados *ADTS, MP3*. El manifiesto también contiene meta-data y está en formato *M3U8* una extensión de lista de reproducción. El contenido publicado se referencia por los clientes vía manifiesto.

*HLS* funciona bien para contenido *LIVE/VOD* indistintamente. Para el caso de *LIVE* se requieren además un codificador multimedia previo en el proceso. A diferencia de los casos anteriores no es necesario módulos especiales para entregar el contenido a los clientes. No hay en este caso por el lado de los servicios ningún módulo especializado para la distribución, un servicio común de *HTTP* es suficiente para la entrega del contenido, ya que este está previamente codificado en archivos independientes para cada segmento.

El codificador multimedia toma una señal en tiempo real de un dispositivo de audio/video, codifica el contenido y lo encapsula para ser transportado. La codificación debe ser en un formato soportado por los dispositivos cliente, H.264 para video y HE-AAC para audio. El codificador distribuye el contenido multimedia en un flujo de transporte *MPEG-2* sobre la red local al segmentador de flujo. El codificador no cambia las configuraciones del flujo en el medio de la codificación como las dimensiones del video o el tipo de *CODEC*. En este paso se generan las diferentes representaciones a distribuir.

### 4.4.2 Codificado y Publicación

El segmentador de flujo es un proceso que lee el flujo de transporte de la red local y lo divide en una serie de pequeños archivos multimedia de similar duración en tiempo de contenido. Aunque cada segmento está en un archivo independiente, los archivos de video son generados a partir de un flujo continuo que puede ser reconstruido.

El segmentador a su vez crea un archivo de índice que contiene referencias a los archivos multimedia individuales. Cada vez que el segmentador termina de generar un nuevo archivo multimedia, se actualiza el archivo de índice. El índice es usado para rastrear la disponibilidad y ubicación de los archivos multimedia. El segmentador también puede encriptar cada segmento y crear un archivo de clave como parte del proceso. Los segmentos multimedia son guardados como archivos .ts de flujos tipo *MPEG-2* y los archivos de índice son guardados como archivos *.M3U8* una extensión del formato .m3u usado para listas de reproducción MP3.

A continuación en la Figura 4.8, se muestra un ejemplo de un archivo de índice *M3U8* producido por el segmentador para tres representaciones de contenido en donde cada uno vincula a una lista de archivos multimedia de 15 segundos cada uno para cada flujo.

El archivo de índice también puede contener URLs de archivos de clave de encriptación o alternar archivos de índice para la localización de las representaciones.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:7794
#EXT-X-TARGETDURATION:15
#EXT-X-KEY:METHOD=AES-128,URI=https://priv.example.com/key.php?r=52s
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="English",
DEFAULT=YES,AUTOSELECT=YES,LANGUAGE="en", URI="main/english-audio.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="Deutsch",
DEFAULT=NO,AUTOSELECT=YES,LANGUAGE="de", URI="main/german-audio.m3u8"
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="aac",NAME="Commentary",
DEFAULT=NO,AUTOSELECT=NO,URI="commentary/audio-only.m3u8"
#EXT-X-STREAM-INF:BANDWIDTH=1280000,CODECS="mp4a.40.5",AUDIO="aac" low/video-only.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2560000,CODECS="mp4a.40.5",AUDIO="aac" mid/video-only.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=7680000,CODECS="mp4a.40.5",AUDIO="aac" hi/video-only.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=65000,CODECS="mp4a.40.5",AUDIO="aac" main/english-audio.m3u8
```

Figura 4.8: Manifiesto de *HTTP Live Streaming*

El segmentador de archivos toma flujos codificados y encapsula en archivos de flujo *MPEG* en unidades de igual longitud en tiempo de contenido.

El segmentador de archivos realiza la misma tarea que el segmentador de flujo con la diferencia que toma archivos como entrada en lugar de flujos. Los archivos de segmentos multimedia son normalmente generados por el segmentador de flujo, basado en la entrada del codificador, y consiste en una serie de archivos .ts que contienen segmentos de flujos de transporte *MPEG-2* que llevan video H.264 y audio AAC.

El sistema de distribución es muy simple, es suficiente con un servidor web o un sistema de cache web que entrega los archivos multimedia y los archivos de índice a los clientes sobre *HTTP*. No es necesario ningún módulo específico en el servidor para poder hacer la distribución del contenido. La configuración recomendada para el servidor se limita a especificar las asociaciones de tipos *MIME* para los archivos *M3U8 (application/x-mpegURL o vnd.apple.mpegURL* y *video/MP2T* archivos tipo .ts.

Configurando los valores de *TTL, Time-To-Live* para los archivos *M3U8* se consigue la conducta de *Cache* deseada, ya que estos archivos se sobrescriben frecuentemente y la última versión debe ser descargada en cada solicitud.

Cuando se quiere desplegar un contenido, el reproductor busca primero el *Index-File* o manifiesto por medio de la URL que lo identifica. Este índice permite localizar los segmentos de video en forma de *HTTP Get* directo para cada archivo correspondientes o sus posibles alternativos en otras representaciones con el fin de bajarlos en forma ordenada para recomponer el flujo original a desplegar. Es el reproductor el que tiene la libertad de elegir la secuencia a descargar para recomponer el flujo original, esto le da la capacidad adaptativa.

El reproductor también baja las claves y realiza el proceso des-encriptado si es necesario. El proceso de bajada se repite hasta que se localiza una etiqueta *#EXT-X-ENDLIST*, si ésta no

se encuentra, entonces el contenido está generándose o es video *LIVE*, en este caso se baja periódicamente el *Index-File* para encontrar las *URLs* de los segmentos nuevos.

### 4.4.3 Cifrado

*HTTP Live Streaming* también proporciona soporte para encriptación y autenticación de usuario sobre *HTTPS* para que los proveedores de contenido puedan protegerlo. La especificación del *HTTP Live Streaming* está en análisis por *Internet Engineering Task Force IETF* para su estandarización [24].

## 4.5 Dynamic Adaptive Streaming HTTP (DASH)

*Dynamic Adaptive Streaming over HTTP* más conocido como *DASH* es una tecnología desarrollada recientemente por *MPEG*, para fijar un estándar de *HAS*, ISO/IEC 23009-1 que se puede descargar de http://standards.iso.org/ittf/PubliclyAvailableStandards/index.html. *DASH* está diseñado para ser independiente de la codificación usada, pero la especificación recomienda el uso de *MPEG-4* en archivos o *MPEG-2* [35].

### 4.5.1 Componentes del servidor

*MPEG DASH* proporciona mecanismos que estandarizan y podrían permitir a futuro la compatibilidad entre los esquemas propuestos por los diferentes actores del mercado. Sin embargo la especificación no requiere un *CODEC* particular, cuando se habilite un servicio de *Streaming*, no hay compatibilidad garantizada por este motivo, los clientes que consumen un servicio *MPEG DASH* tendrían que ser capaces de manejar el *CODEC* con el que se generó el contenido a fin de mantener la compatibilidad. Estas cuestiones han frenado la adopción con respecto a otras arquitecturas de *Streaming* [25].

### 4.5.2 Codificado y Publicación

En la Figura 4.9 se puede observar que igual que otras tecnologías de *Streaming* de tipo adaptativo, el cliente solicita un manifiesto XML. En la Figura 4.10 se puede ver un ejemplo de manifiesto, llamado Descripción de Presentación para Medios *MPD* [36]. Este manifiesto, de tipo *VOD ISO Base File Format*, contiene información acerca de flujos para audio en dos calidades e idiomas diferentes. También contine video *Streams* en sus distintas representaciones, localizables en dos orígenes alternativos *CDN1/CDN2*.

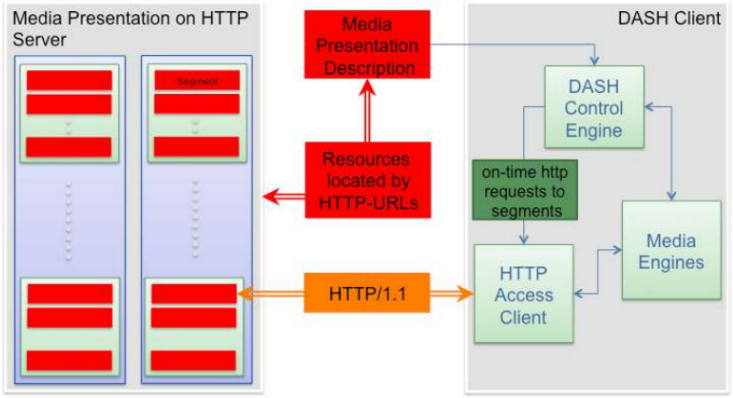


Figura 4.9: Funcionamiento

### 4.5.3 Cifrado

El *MPD* también puede enumerar varias representaciones diferentes del contenido para las distintas arquitecturas existentes, aunque cada plataforma puede tener distintas necesidades de *CODEC*. La especificación *DASH* no dicta un esquema común de cómo se debe implementar el cifrado, por lo que las plataformas cliente deben tener también los diferentes procesos DRM, necesarios para descifrar y decodificar el contenido.

Al igual que las especificaciones propietarias, permite el agregado de otros flujos de información además del audio/video al contenido. Como característica particular se habilita también la posibilidad del envío de eventos para que sean interpretados en el cliente *DASH* según el modelo de la Figura 4.11.

## 4.6 Resumen

Se resalta en el Cuadro 4.1, las principales propiedades de las Arquitecturas, mostrando las similitudes y diferencias.

Los esquemas adaptativos tienen las siguientes ventajas en la distribución de contenido.

- Reducen el costo de distribución, utilizando la infraestructura de *Cache* existente en Internet.
- No tienen inconvenientes para atravesar *Firewalls*, ya que utilizan *HTTP* como protocolo de transporte.
- Son funcionales tanto para *Streaming VOD* como *LIVE* en modalidad adaptativa.
- Se adaptan dinámicamente a las capacidades de la red.

```
<?xml version="1.0" encoding="UTF-8"?>
<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 DASH-MPD.xsd"
 type="static"
 mediaPresentationDuration="PT3256S"
 minBufferTime="PT1.2S"
 profiles="urn:mpeg:dash:profile:isoff-on-demand:2011">
 <BaseURL>http://cdn1.example.com/</BaseURL>
 <BaseURL>http://cdn2.example.com/</BaseURL>
 <Period>
  <!-- English Audio -->
  <AdaptationSet mimeType="audio/mp4" codecs="mp4a.40" lang="en" subsegmentAlignment="true"
   subsegmentStartsWithSAP="1">
   <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
   <Representation id="1" bandwidth="64000">
   <BaseURL>7657412348.mp4</BaseURL>
   </Representation>
    <Representation id="2" bandwidth="32000">
    <BaseURL>3463646346.mp4</BaseURL>
   </Representation>
   </AdaptationSet>
   <!-- French Audio -->
   <AdaptationSet mimeType="audio/mp4" codecs="mp4a.40.2" lang="fr" subsegmentAlignment="true"
   subsegmentStartsWithSAP="1">
   <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
   <Role schemeIdUri="urn:mpeg:dash:role:2011" value="dub"/>
   <Representation id="3" bandwidth="64000">
    <BaseURL>3463275477.mp4</BaseURL>
   </Representation>
   <Representation id="4" bandwidth="32000">
    <BaseURL>5685763463.mp4</BaseURL>
   </Representation>
   </AdaptationSet>
   <!-- Timed text -->
   <AdaptationSet mimeType="application/ttml+xml" lang="de">
    <Role schemeIdUri="urn:mpeg:dash:role" value="subtitle"/>
    <Representation id="5" bandwidth="256">
    <BaseURL>796735657.xml</BaseURL>
   </Representation>
   </AdaptationSet>
   <!-- Video -->
   <AdaptationSet mimeType="video/mp4" codecs="avc1.4d0228" subsegmentAlignment="true"
   subsegmentStartsWithSAP="2">
   <ContentProtection schemeIdUri="urn:uuid:706D6953-656C-5244-4D48-656164657221"/>
   <Representation id="6" bandwidth="256000" width="320" height="240">
   <BaseURL>8563456473.mp4</BaseURL>
   </Representation>
    <Representation id="7" bandwidth="512000" width="320" height="240">
    <BaseURL>56363634.mp4</BaseURL>
   </Representation>
   <Representation id="8" bandwidth="1024000" width="640" height="480">
    <BaseURL>562465736.mp4</BaseURL>
   </Representation>
    <Representation id="9" bandwidth="1384000" width="640" height="480">
    <BaseURL>41325645.mp4</BaseURL>
   </Representation>
    <Representation id="A" bandwidth="1536000" width="1280" height="720">
    <BaseURL>89045625.mp4</BaseURL>
   </Representation>
   <Representation id="B" bandwidth="2048000" width="1280" height="720">
    <BaseURL>23536745734.mp4</BaseURL>
   </Representation>
  </AdaptationSet>
 </Period>
</MPD>
```

Figura 4.10: Manifiesto *MPD DASH*

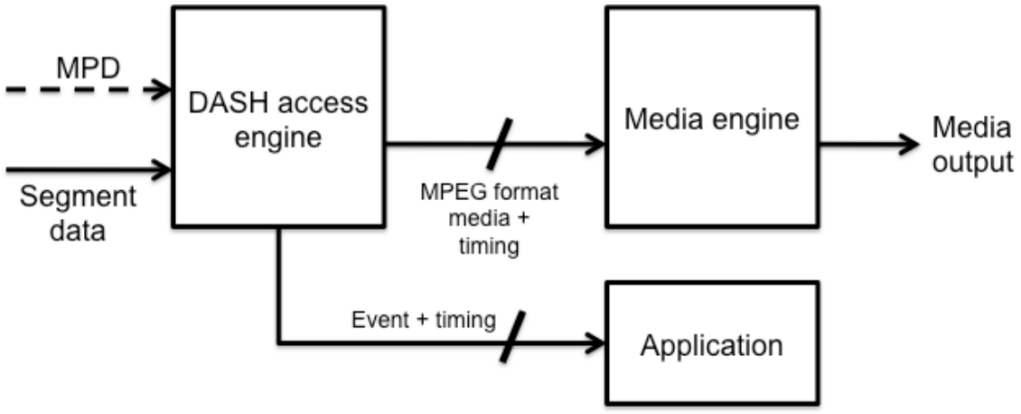


Figura 4.11: Extensiones del manifiesto al procesamiento de eventos en el cliente *DASH*

| | SmoothStreaming | HLS | HDS | DASH |
|---|---|---|---|---|
| Contenedor | MP4 | MP2 TS | MP4 | MP4,MP2 TS |
| Storage | File-x-Rate | File-x-Chunk | File-x-Rate | File-x-Chunk |
| Segment seg. | 2 | 10 | particular | flexible |
| Data File | .isma, ismv | .ts | .f4f | .mp4,.m4s |
| Manifest | .ism | .m3u8 | .f4x | .mpd |
| Client | .ismc | .m3u8 | .f4m l | .mpd |
| CODEC | H.264-VC1 AAC-WMA | Mpeg4,H.264 AAC-mp3 | H.264-VP6 AAC-MP3 | Open |

Cuadro 4.1: Principales propiedades de las Arquitecturas *HAS*.

Sin embargo hay sutiles diferencias entre ellos.

- Microsoft y Adobe requieren software particular en los servicios para entregar los segmentos en contraste al esquema de Apple, donde solo se usa servidores de *HTTP* estándar con manifiestos índice.
- MS y Adobe usan MPEG4 (MP4) File Format para el almacenamiento y el transporte, de video en segmentos. Apple utiliza MPEG2 Transport Stream File Format para enviar el contenido fragmentado almacenado en sus servicios. MPEG2 tiene menos rendimiento en el transporte de información útil, en comparación con MPEG4, pero es un formato más sencillo de consumir (requiere menos CPU, fue pensado originalmente en tiempos en que este recurso era escaso).
- En cada caso la protección del contenido se hace en forma integrada y diferente para cada sistema.

### 4.6.1 Preparación de Contenidos en general

Todas estas tecnologías adaptativas, requieren un esfuerzo computacional intenso para generar las representaciones necesarias en diferentes codificaciones, (en particular para el contenido LIVE donde el horizonte temporal es más pequeño y el contenido está generándose). Sin embargo este proceso es independiente de la cantidad de consumidores que tenga el servicio de video. La escalabilidad de este tipo de solución es factible mediante los esquemas de replicación ya existentes en las CDNs sobre HTTP.

Inicialmente *HTTP Live Streaming* propuesto por Apple, difundía contenidos solo para sus dispositivos *iPad/iPhone*, luego aparecieron otros consumidores para contenido presentado en esta modalidad de *Streaming*. Cada fragmento es un archivo separado para estas arquitecturas, que se almacena en los servicios, por lo que es fácil de implementar, pero al mismo tiempo cuestiona su escalabilidad a medida que el número de fragmentos aumenta.

Microsoft y Adobe apuntan a difundir contenido a dispositivos o reproductores en forma general, utilizan un esquema de manifiesto plano en formato XML. En la preparación del contenido ambos utilizan un mecanismo similar, se generan manifiestos de archivos indexados. Módulos de servidor de *Smooth Streaming* o *HTTP Streaming* entregan los segmentos a partir de desplazamientos en el archivo codificado.

*DASH* se propone como un nuevo estándar para proporcionar interacción entre las plataformas y permite agregados de meta-data adicionales al contenido, tales como idiomas, u otra información extra a desplegar. La codificación del contenido y *DRM* debe hacerse según se requiera para cada destino repetidamente.

### 4.6.2 Problema General

En todos los casos es necesario utilizar una política en la selección de la secuencia de segmentos a descargar, para re-ensamblar el contenido a desplegar. Esto tiene alto impacto en la calidad: si se elige una baja codificación entonces se consumen menos recursos de red pero la calidad no es mejor; si se aumenta la codificación mejora la calidad pero aumenta la probabilidad de *re-buffering*. Al fluctuar la red, no existe una mejor solución que perdure en el tiempo, por tanto debe ajustarse continuamente. Por otro lado repetidos cambios de *bit-rate* son percibidos por el usuario como deterioro de calidad. En adelante se enfoca este problema y las soluciones actuales para cada arquitectura.

# Parte II

# IMPLEMENTACIONES DE REFERENCIA

# Capítulo 5

# Componentes Principales

En la introducción de este caso de estudio se deja planteado cual es el problema general en el que se trabajará. En este capítulo se presentan cuáles son las factores que afectan el funcionamiento de la arquitectura HAS y por que se necesita determinar una codificación para los segmentos de video descargar.

## 5.1 Diagrama de Bloques

En la Figura 5.1 se presenta un diagrama de bloques de un sistema reproductor de video típico [13], desde un enfoque general, se observa la interacción entre los módulos. El sistema se conecta a una red de datos mediante un *Communication Module*, que le permite obtener los segmentos de video según lo requerido por el módulo *Heuristic*. Los segmentos son procesados, en forma secuencial, filtrando los datos en audio y video, para componer en *UI Rendering* un contenido desplegable que es la salida del sistema.

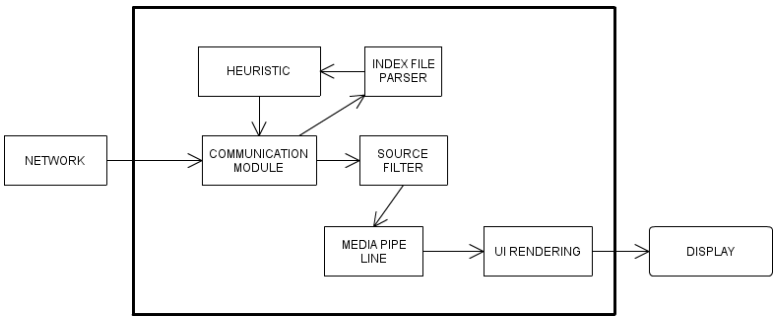


Figura 5.1: Módulos principales de un reproductor de video *HAS*, donde y como interactúa la Heurística [13].

Las heurísticas insertas en los reproductores eligen la secuencia de segmentos a ensamblar en un flujo que luego será desplegado. Persiguen el objetivo de maximizar la calidad percibida

y se alcanzaran valores altos cuando el contenido desplegado sea lo más fiel a la señal original posible. Las transferencias de las señales de video codificadas como datos, no son ideales y acumulan errores a lo largo de todo el proceso.

En el momento de la codificación del video, los parámetros son la resolución horizontal-vertical, número de cuadros por segundo (*Frame Rate*) y la taza de bits (*Bit-Rate*). Estos permiten controlar el error en la representación. Donde representaciones más fieles, generan mayor volumen de datos o segmentos más grandes para mismos intervalos de contenido.

Los errores que aparecen finalmente en la salida de un sistema HAS, se introducen en el proceso de codificación o en las transferencias de datos. Los procesos de codificación son controlables y permiten obtener una representación en la calidad deseada, mientras que otros fenómenos son menos predecibles.

La degradación en las transferencias de los datos impactan en el sistema, en algunos casos hasta su detención. La detención ocurre cuando el reproductor se queda sin datos para consumir. Esto es: algunos de sus *Buffers* de datos se vacía. Esto se verá en detalle en el Capítulo 6.

## 5.2 Jitter

Durante el *Video Streaming HTTP* ocurren variaciones en el comportamiento del sistema en su conjunto (servicio–CDN–reproductor) visto en 2.1, donde aparecen pérdidas en los servicios al momento de la solicitud, así como en el propio funcionamiento de la CDN. Estas inestabilidades se transfieren al reproductor en forma de un retraso variable en la llegada de los datos.

Si bien *HTTP* asegura la transferencia de la información sin alteración, por estar basado sobre *TCP/IP*, no se enfoca en cuanto tiempo se necesita para la descarga, es decir existe una variabilidad importante en el proceso. Estas variaciones introducen problemas que afectan al reproductor en la disponibilidad de los datos para mantener el flujo de consumo continuo.

El propio mecanismo de control de congestión de *TCP*, genera un patrón variable en el tamaño de la ventana *W* de transmisión (cantidad de paquetes máximo que se envían antes de una confirmación), esto desencadena un comportamiento de ráfagas en la llegada de la información. En la Figura 5.2 se muestra un modelo para cuantificar la transferencia o rendimiento de *TCP* propuesto por [20], en función de variables promedio relevantes, donde, *p* es la probabilidad simple de pérdida de un paquete, *(MSS), Maximum Segment Size* y *(RTT), Round-Trip Time*.

$$T = \frac{MSS \times C}{RTT \times \sqrt{p}}$$

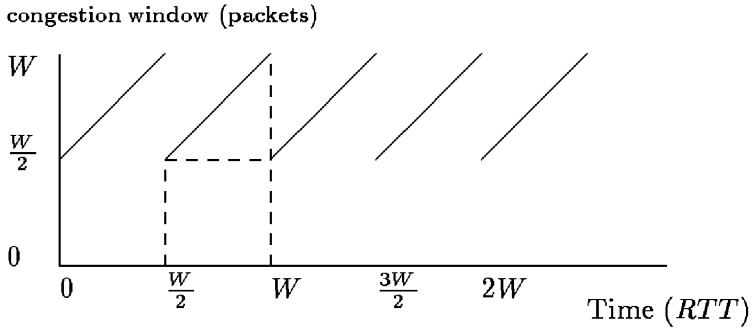


Figure 1: TCP window evolution under periodic loss
Each cycle delivers $(\frac{W}{2})^2 + \frac{1}{2}(\frac{W}{2})^2 = 1/p$ packets and takes $W/2$ round trip times.

Figura 5.2: Rendimiento de conexión tipo TCP, propuesto por [20]

$$C = \sqrt{\frac{3}{2}}$$

En forma independiente pero con el mismo efecto en la variabilidad del tiempo de la descarga, está el comportamiento de los compresores o *CODECs* de video. Aún dentro de una misma calidad de representación el tipo de contenido a transportar genera segmentos de dimensiones muy distintas. Dependiendo del tipo de contenido en la señal codificada, es necesario más información para escenas de acción, que para contenidos estáticos. Diferencias de tamaño de entre 30 % y 50 % no son extrañas solo por causa del contenido.

En la Figura 5.3 se muestra como afectan las causas anteriores en la llegada a tiempo de la información. Cada segmento de video representa un contenido con una duración definida. En un ensayo previo se busca poner en relevancia como afecta la probabilidad de llegada a tiempo de la información. Se compara el tiempo requerido para la descarga, con la duración de su propio contenido. En este ejercicio una probabilidad de detención de 1, implica no mantener la continuidad.

Asumiendo que los segmentos son equivalentes en duración de contenido, y las descargas son en secuencia, el siguiente estará disponible si el tiempo de descarga es menor a su duración. Esta variabilidad afecta la continuidad del *Stream*.

Se utiliza una plataforma de pruebas similar a la usada en Capítulo 9.4 y se fija una transferencia emulada con retraso de 150 +/- 50ms y 0.5 % pérdidas, valores similares a los encontrados en la realidad. El efecto de estas pérdidas se traduce a nivel de HTTP, en un ancho de banda variable, para la descarga de los segmentos. Todas estas variaciones generan inestabilidades en el sistema de video *Streaming HTTP* y se transforman en demoras impredecibles, en algunos casos comparativamente muy largas o equivalentes al cierre del flujo, con consecuencias des-

agradables para la experiencia del usuario.

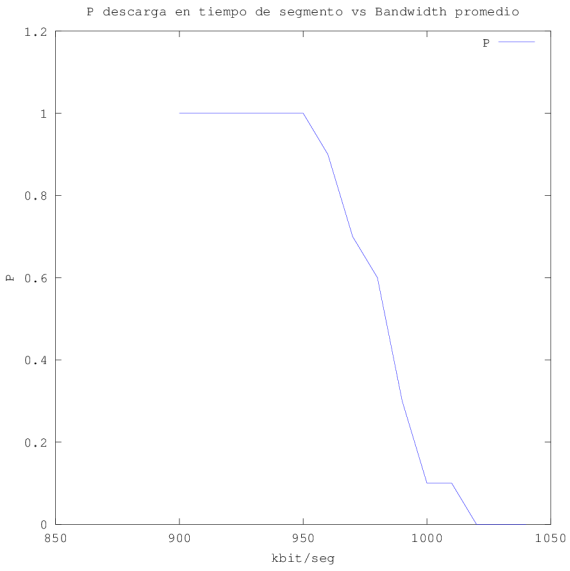


Figura 5.3: Probabilidad de descargar el segmento de video en menos tiempo que la duración de su representación.

# Capítulo 6

# Heurísticas para seleccionar calidad

Con el fin de atenuar los impactos de la variabilidad que afectan al sistema, se aplican con éxito técnicas de *Buffering* en general. Se puede describir un *Buffer* como un proceso *FIFO, Firt In First Out* para los datos que almacena. Este mecanismo recibe la información en ráfagas provocadas por las variabilidades vistas.

La ventaja que da, es hacer posible un consumo constante de datos a partir del *Buffer*, para las etapas finales del desplegado. En general hay dos almacenamientos de este tipo en los reproductores, uno para la descarga de segmentos *Download Buffer* y otro para colocar el contenido a mostrar *Video Buffer*. En la Figura 6.1 se ve donde están ubicados, según el diagramas de componentes del Capítulo 5.1.

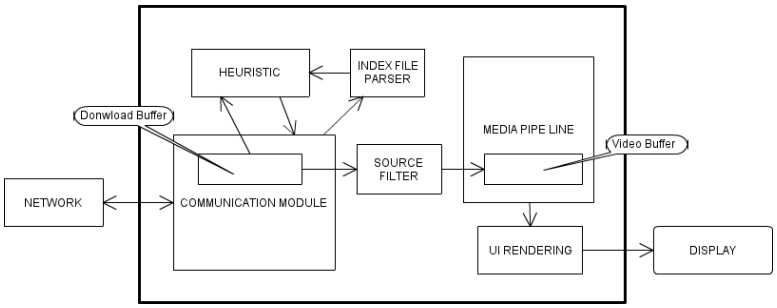


Figura 6.1: Componentes relevantes para la operación de la heurística. Los módulos *Comunication* y *Pipeline* incluyen cada uno un *Buffer*.

Para el problema general planteado en 4.6.2, la elección del tipos de descarga, debe conseguir mantener niveles aceptables en estos almacenamientos intermedios, con el fin de mantener un desplegado continuo del *Stream* de video.

## 6.1 Heurísticas de selector de segmentos

La heurística de cada reproductor determina que secuencia de segmentos se van a descargar como mecanismo de adaptación a las variabilidades. Las heurísticas revisadas a continuación, relacionan fuertemente aspectos métricos del rendimiento en las transferencias de datos, proponiendo la secuencia de segmentos a descargar. Todas, utilizan como criterio, que una mejor representación en la codificación de la señal original, causará una mejor experiencia en el despliegue para el usuario [3].

*HAS* establece un largo predeterminado en tiempo de contenido para el segmento, que se mantiene para todas las representaciones. Esto permite un control gradual sobre la cantidad de información a descargar, con el fin de favorecer la continuidad en el flujo de datos, frente a la exactitud del desplegado con respecto a la señal original. En general la heurística tiene tres objetivos:

1. En la apertura del *Stream*, las heurísticas tienen como objetivo principal obtener un *Buffer* con suficiente rapidez para iniciar el desplegado.

2. Luego de iniciado el desplegado, mantener el flujo de datos estable, para evitar el agotamiento del *Buffer* y por lo tanto detenciones.

3. Intentar desplegar la representación de mayor calidad posible.

A continuación veremos para los reproductores de cada arquitectura, las heurísticas y los criterios que usan.

## 6.2 Microsoft Smooth Streaming

### 6.2.1 Bloques en el reproductor

Como propuesta inicial para el control adaptativo, *Microsoft* presenta para la heurística del reproductor un esquema con los siguientes bloques principales: *Dowloader*, *Index File Parser*, *Heuristics* [13].

El *Downloader* administra la conexión con los servicios y encola solicitudes de segmentos a descargar en forma de *URLs* que se solicitan de acuerdo a la representación requerida. Cuando el sistema inicia, el manifiesto es requerido antes que nada. La información que proporciona el *Index-file Parser* hace posible la planificación de las tareas del *Downloader*, restringido al resultado de la heurística, que hace la selección de la representación del contenido a descargar.

El bloque de la heurística tiene como objetivos:

- Arranque rápido del desplegado de contenido o fase de *Start-up*.

- Adaptar la representación del contenido que mejor se adapta al BW disponible.

- Adaptar la representación del contenido a la capacidad existente en el reproductor.

Se desarrolla una actividad secuencial que descarga los segmentos seleccionados, componiendo un media-pipeline en un *Buffer* que permite desplegar el video. Las principales variables evaluadas de forma local y tomadas como datos de entrada para la selección de la representación a descargar son:

- Valor de *Bandwidth* actual

- Tamaño del *Buffer* actual

- Capacidades del dispositivo

- Formato del visualizador

Esta heurística intenta determinar, a largo plazo (siempre hacia delante para un número prefijado de segmentos), si el *Buffer* mínimo se alcanzará o no. En el caso de existir la posibilidad de agotamiento, según la proyección actual de las condiciones a futuro, entonces se hace necesario hacer cambios en la política de descarga, con el fin de superar la cota mínima como sugiere la Figura 6.2. Fijando descargas en representaciones más adecuadas, se logra mantener los niveles del *Buffer* dentro de una zona de seguridad.

Al comienzo del desplegado hay poca información sobre el comportamiento de las descargas, por lo que inicialmente se intenta obtener representaciones bajas. Con este criterio es posible abrir el flujo rápidamente y comenzar el desplegado.

### 6.2.2 Mecanismo general de la Heurística

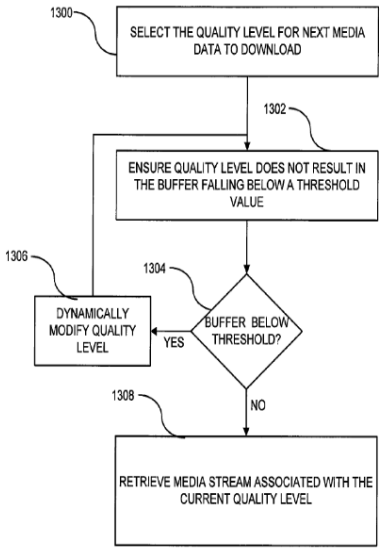


Figura 6.2: Bloques principales según publicación [13].

La heurística verifica, con un ensayo a valores promedio actuales, si es posible que en tiempo futuro aún sea posible mantener el flujo de datos para un desplegado sin detenciones.

La inferencia es una proyección de las condiciones actuales, locales al sistema del reproductor, principalmente el *Bandwidth* de la conexión. La decisión por la afirmativa es mantener el mismo tipo de representación que se usó para la descarga anterior. Por la negativa se aplica un algoritmo, que tiene como resultado cual es la nueva representación conveniente, ensayando una predicción a futuro del estado del *Buffer*. En la Figura 6.3, la función *PredictBuffer* se muestra como algoritmo adjunto.

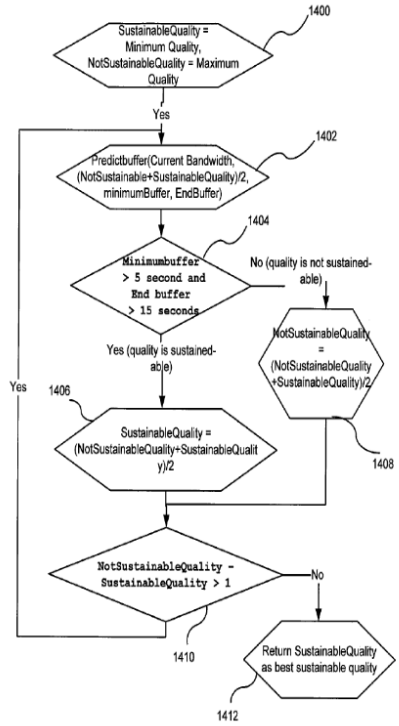


Figura 6.3: Selección de próxima representación a descargar. Propuesto como ejemplo en [13].

```
Function PredictBuffer(_in Bandwith, _in ProposedQuality, _out minimimbuffer, _out endbuffer){
        endbuffer = Current buffer size
        mimimunbuffer = endbuffer
        for (chunckindex = currentindex to next 60 chunks){
                Scan all streams for chunkindex, find the chunk with nearest video quality to ProposedQuality
                Endbuffer = endbuffer + (chunkduration - (chunksize/bandwitb))
                If (endbuffer < minimumbuffer )
                        Minimunbuffer = endbuffer
        }
}
```

Figura 6.4: Función *PredictBuffer*.

Con el resultado de la función de la Figura 6.4, el *Code-Rate* preferido, queda entre cotas que se van ajustando a medida que pasa el tiempo y se promedian las condiciones evaluadas localmente.

## 6.3 Adobe's Dynamic HTTP Streaming

*JWPlayer* es un Reproductor de video de código abierto que está desarrollado a partir del *FrameWork* de Adobe que utiliza tecnología *Dynamic Streaming*.[1]

Internamente la heurística opera por topes (por la aplicación de reglas). Las variables al comienzo toman valores por defecto, que permiten una fase inicial, incentivando la descarga de segmentos, uno tras otro en forma secuencial. Al mismo tiempo aparece información sobre la evolución de las descargas que permite aplicar reglas para un mejor ajuste.

Esta heurística comienza cuando un reproductor de video selecciona un *Stream* para consumir, descarga el índice o manifiesto que relaciona el contenido con los archivos que lo codifican en las representaciones que se ofrecen.

Luego comienza la bajada de segmentos en forma secuencial hasta alcanzar un mínimo, en tiempo de contenido de 12 segundos disponibles en el *Video Buffer* local. En todo momento se trata de mantener un 100 % del *Buffer* completo o pedir nuevos segmentos para mantenerlo.

Si el contenido del *Video Buffer* cae por debajo de 25 % , se produce una detención del desplegado y entra en estado de *Re-Buffer*, cuando se alcanza el 25 % se cambia a estado *Play* y continúa.

Si hay un movimiento de desplazamiento se descarta todo el contenido y todo se considera como al principio. Con el fin de alcanzar el desplegado rápidamente, se cambia a la codificación más baja declarada en el manifiesto, esto se llama *Fast-Start* o arranque rápido.

Desde el inicio se calcula cual es la taza de transferencia, esta información se hace consistente a medida que ocurren las descargas. La sección que propiamente selecciona la representación del próximo segmento está dentro de la función *updateLevel()* que es ejecutada de 8 a 10 veces por segundo.

### 6.3.1 Comportamiento

Como se ve en la Figura 6.5, se trata de asignar el nivel más alto posible considerando directamente el ancho de banda como selector del tipo de representación a descargar.

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

```
for(var j:Number = _levels.length - 1; j > 0; j--) {
        if( _levels[j].bitrate <= _bandwidth * BITRATE_FACTOR {
            level = j;
            break;
        }
    }
```

Figura 6.5: Determinación de nivel de representación en función de ancho de banda disponible.

A continuación se ve en la Figura 6.6 como se restringe el comportamiento anterior, para el ascenso, de un nivel por vez.

```
if(level != _level) {
        if(level > _level) {
                _level++;
        } else {
                _level = level;
        }
}
```

Figura 6.6: Restricción para el ascenso de nivel de representación.

Esto da lugar a patrones de comportamiento típicos observados para esta heurística, de modo que cuando ocurre una degradación importante del *Bandwidth*, hace que se re-inicie el proceso de ascenso desde una representación muy baja. En la Figura 6.7, se ve un patrón escalonado en el tipo de representaciones que se descargan, una vez que sucede la detención marcado por la línea llena inferior.

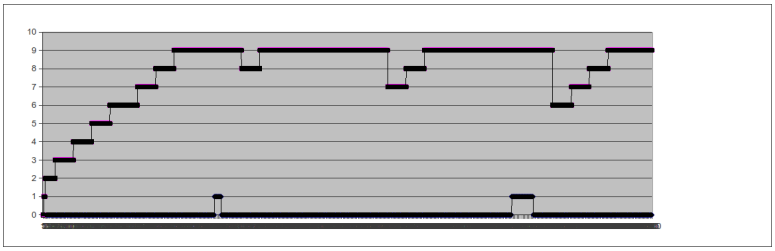

Figura 6.7: Comportamiento típico, cuando hay un vaciado del *Buffer* de video.

## 6.4 Apple's HTTP Live Streaming

### 6.4.1 Comportamiento

Para *HLS* el reproductor busca el archivo índice o manifiesto apuntado por una *URL* que identifica al contenido a consumir. El archivo de índice especifica la ubicación de los archivos

multimedia disponibles, las claves de criptografía de los flujos alternativos disponibles. El formato del manifiesto es una lista de reproducción extendida tipo *M3U8*.

Para cada flujo seleccionado, el cliente descarga, cada archivo o segmento disponible. Cada archivo contiene un segmento consecutivo del flujo en formato MPEG-2 ISO/IEC 13818-1. Una vez que se ha descargado una cantidad suficiente de segmentos, el reproductor comienza a presentar el flujo re-ensamblado al usuario.

Apple incentiva que exista al menos un *Stream* básico mínimo de 64Kbps para distribuir audio y video o audio con fotos como mecanismo de mejorar la experiencia en condiciones de anchos de banda mínimos para dispositivos móviles.

La lista de reproducción *M3U8* se puede preparar de manera conveniente para que los servicios tengan una capacidad de *Failover*. Como se muestra en la Figura 6.8, continúa operando a partir de múltiples orígenes. Esto es aplicable tanto para el propio manifiesto como para los archivos o trozos de la lista. Como ejemplo para el caso de dos servidores “ALFA”, “BETA” y múltiples representaciones:

```
#EXTM3U
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000
http://ALPHA.mycompany.com/lo/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=200000
http://BETA.mycompany.com/lo/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000
http://ALPHA.mycompany.com/md/prog_index.m3u8
#EXT-X-STREAM-INF:PROGRAM-ID=1, BANDWIDTH=500000
http://BETA.mycompany.com/md/prog_index.m3u8
```

Figura 6.8: Manifiesto para *HLS* con capacidad de *Failover*.

El fabricante no especifica como funciona el mecanismo de selección entre los flujos a consumir [24]. El comportamiento observado en nuestras pruebas de laboratorio de la Sección 9.3, es una adaptación lenta y conservadora, poco reactiva a cambios rápidos.

## 6.5 Dynamic Adaptive Streaming HTTP

*DASH (Dynamic Adaptive Streaming over HTTP)* conmuta las representaciones del contenido a descargar para adaptar a las condiciones locales en que se encuentra el reproductor de video.

El mecanismo de adaptación, como en los demás casos, no queda definido exactamente en la definición de *DASH*, se han propuesto algunos basados en las condiciones que impone la propia descarga.

### 6.5.1 Evaluación de Descargas

En [16] se revisa el esquema que propone Liu, que llama *RAHS (Rate Adaptation for Adaptive HTTP Streaming)* , que previene el agotamiento de *Buffer* y reduce el cambio frecuente en las representaciones a descargar.

El esquema *RAHS* determina la influencia de las condiciones de red como:

$$\mu = \frac{MSD}{SFT}$$

$SFT$ es el tiempo desde que se emite el *HTTP Get*, hasta que el último bit del segmento es descargado $MSD$ es el tiempo medio de duración en contenido para los segmentos.

Si $\mu$ aumenta, es posible descargar representaciones más fieles a la señala original, si $\mu$ disminuye, es necesario cambiar a codificaciones menores, ya que en promedio no se logrará mantener una condición estable a largo plazo.

Con este criterio la heurística opera incrementando o decrementando el nivel de representación próximo a descargar.

Un esquema similar, el *ASAC (Adaptive Streaming of Audiovisual Content)*, también analizado en [16], propone el cálculo de $SFT$ con el uso de media móvil. Con esto se estabiliza el indicador $\mu$, en forma histórica para las descargas de los segmentos, y al mismo tiempo se pierde reacción del sistema.

*S-DASH* es un esquema que tiende a minimizar el impacto de los cambios de representación a descargar, con la finalidad de no degradar la experiencia de visualización del usuario, por la conmutación en si misma. En este caso se utiliza además de $\frac{MSD}{SFT}$, otras métricas como *Segment Throughput Measurement*, en el cálculo de la adaptación, pero siempre basado en el estado del *Buffer* y en capacidad que permite el medio para conseguir segmentos.

El algoritmo propuesto capta la variación en el rendimiento de las transferencias y estima mejor la elección, para mantener la continuidad con holgura.

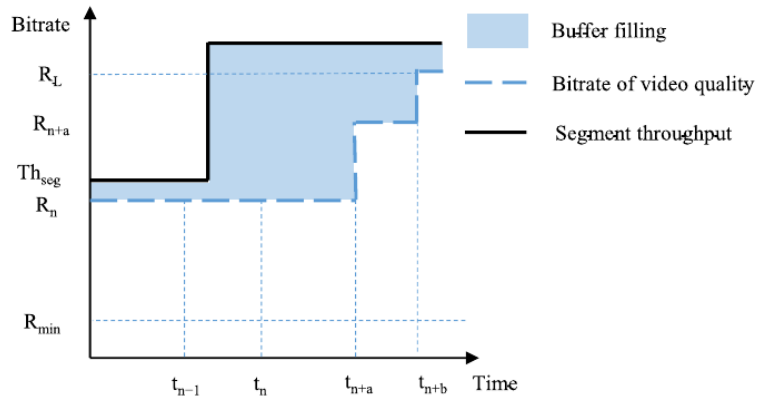


Figura 6.9: Para el ascenso en niveles de representación.

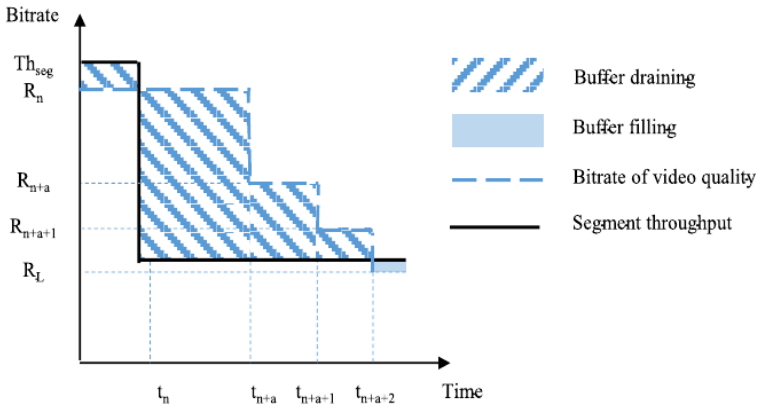


Figura 6.10: Para el descenso en niveles de representación.

Mediante la aplicación de reglas se opera sobre el momento en que se realiza la conmutación de representación. En la Figura. 6.9 se muestra como a pesar de un cambio ascendente en el rendimiento de la red en $t_n$, se ejecuta el cambio en $t_n + a$, con el fin de prevenir el agotamiento de *Buffer*. El descenso se hace de a pasos, como muestra la Figura. 6.10 con la finalidad de mantener la holgura y suavizar los saltos de representación.

## 6.6 Otros ejemplos

Muchos análisis parten del concepto anterior, esquema de Liu [40], mientras que algunas propuestas independientes, como por ejemplo *ENET* [39], sigue la idea de análisis de *Buffer*.

Otra propuesta de algoritmo similar usado en la elección del tipo de representación de segmentos para *HAS* es *Elastic*, propuesto por [11], usando indicadores del proceso de descarga,

como % de tiempo que no se usa para descargas (indicador de que es posible obtener una mejor representación), ver Figura 6.11.

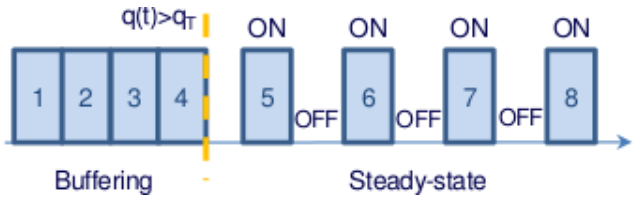


Figura 6.11: Estados y % tiempo para descargas.

## 6.7 Resumen

Se ha visto con varios ejemplos como se aplican distintos criterios para abordar el mismo problema general, ”La elección de tipo de representación para segmentos de video, que se deben descargar”, definido en la Sección 4.6.2.

Como se plantea al inicio, *HAS* habilita una capacidad de adaptación como mecanismo para atenuar los problemas de continuidad, lo que se consigue con opciones de descarga de segmentos en diferentes representaciones. Los segmentos que contienen menos información son más fáciles de obtener, pero representan con dificultad la señal original, y también dan una experiencia baja en el desplegado.

Este compromiso de mantener la continuidad, frente a exactitud de la señal original, impacta en la calidad percibida. De acuerdo a lo visto en esta sección hay muchas maneras y criterios diferentes de atacar el problema general.

Las Heurísticas en *HAS* permiten de manera controlada degradar el desplegado a tasas conocidas de error, respecto a la señal original, en favor de mantener la continuidad. Desplegar representaciones menos fieles a la señal original, da la posibilidad de atenuar el problema crítico de la detención.

Mantener la continuidad y desplegar la mejor representación de la señal original disponible, son factores contrapuestos en este sistema. Los tipos de descarga diferentes para un mismo segmento, favorecen a uno u otro factor, ambos impactan en la calidad percibida.

El comportamiento de algunos esquemas tiende a ser conservador al momento de optar por una representación más fiel del contenido, con el fin de mantener un *Buffer* con holgura que permita el desplegado continuo.

El sistema de *Streaming* en su totalidad, debe dar la mejor experiencia en el desplegado del contenido. En lo que sigue para este caso de estudio se resalta la importancia de maximizar

durante el funcionamiento, la calidad percibida. Para esto se verá que es y como se cuantifica la experiencia del desplegado de un video en forma general para esta tecnología.

# Parte III

# CALIDAD PERCIBIDA

# Capítulo 7

# Métricas de Calidad de Video

En este capitulo se describe, desde el punto de vista del usuario, temas relativos a la calidad que se percibe del contenido desplegado. Se resalta por un lado la dificultad que existe en conseguir una referencia representativa del valor real y la utilidad de contar con un indicador que la cuantifique. Se muestran algunos de los métodos para conseguir indicadores de calidad en la experiencia para *Video Streaming*.

La señal de video original se degrada desde su origen acumulando efectos en su recorrido, errores en el proceso de codificación y pérdidas que se introducen por problemas del servicio, fallas en la *CDN* o en el transporte de los datos hasta el cliente.

## 7.1 Mediciones de Calidad Percibida

Las mediciones de calidad percibida por el usuario, están orientadas a obtener una evaluación general del desplegado. Esta cuantificación es muy ventajosa y la más próxima al usuario que se puede obtener para evaluar el desempeño del sistema completo. Los reproductores HAS en funcionamiento conmutan de un tipo de representación a otro, con el fin de lograr adaptación, interesa ver como este comportamiento influye en la percepción de los observadores.

Comenzando por trabajos relativos a la calidad percibida para la plataforma *YouTube*, previo a las tecnologías adaptativas, se muestran los resultados de [14]. En la Figura 7.1 se detallan resultados de la recolección de datos vía *Crowdsourcing*, acerca de como evolucionan las descargas. En escenarios de *HTTP Progresive Download* se citan como causas principales de baja experiencia para el usuario: las detenciones y la duración de las mismas.

Efectos similares ocurren para las tecnologías adaptativas. En la Figura 7.2, se muestran reportes actuales [7] donde aparece la vinculación que existe entre la calidad de servicio del acceso a los datos y la experiencia de los usuarios. Si la experiencia baja, el interés o la permanencia del observador frente al contenido disminuye, como muestra la Figura 7.3.

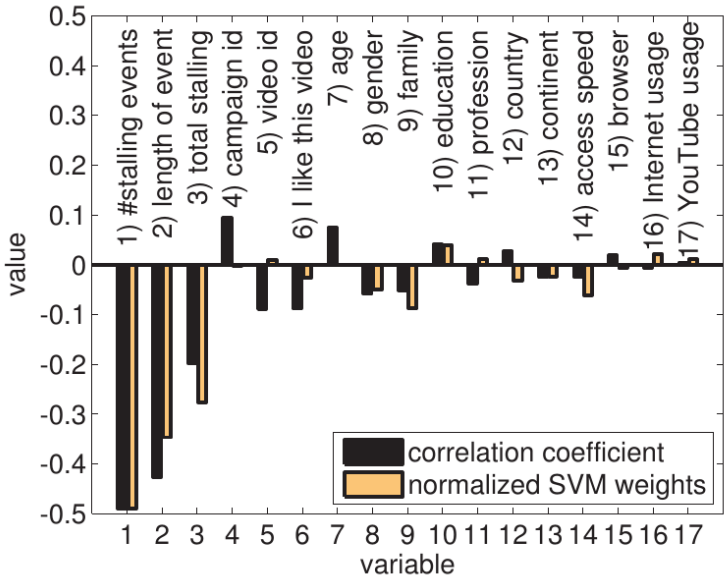


Figura 7.1: Comparación de causas que afectan QoE en *HTTP Progresive Download* [14].

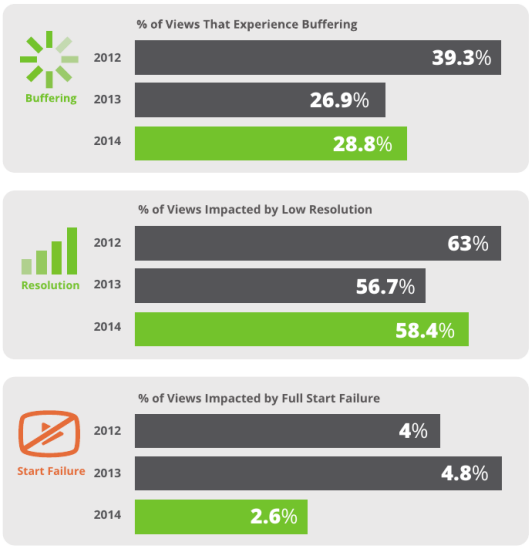


Figura 7.2: Impacto de la calidad de servicio sobre la calidad de la experiencia de los usuarios [7].

En el caso de HAS, el propio funcionamiento del esquema, genera impactos negativos en QoE cuando ocurren frecuentemente cambios o *Switching* entre representaciones del contenido. Estos impactos son: por un lado, se consigue favorablemente superar o mejorar la experiencia general frente a las detenciones; y por otro, cuando hay perfiles de conmutaciones entre

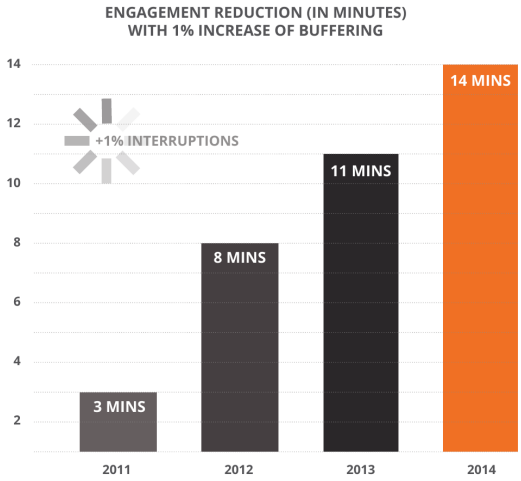


Figura 7.3: Impacto de la experiencia sobre la permanencia de los usuarios viendo el contenido [7].

representaciones poco convenientes, se impacta negativamente en el *MOS*.

En estudios similares se observa que las conmutaciones del sistema HAS producen alteraciones en el *MOS* asociado [28]. Las conmutaciones son el resultado de aplicar las heurísticas sobre el sistema. El objetivo que persiguen en todos los casos, es evitar el congelamiento del desplegado, causa principal de una baja experiencia para el usuario.

La heurística de los reproductores se ejecuta a intervalos discretos, cuando es posible descargar un nuevo segmento al *Buffer* local. Sin embargo en casos particulares se generan perfiles en la selección que terminan afectando a la calidad percibida.

Este tipo de perfiles observados [27], tienen cambios con más frecuencia de la necesaria y terminan dando como resultado una calidad percibida menor, en comparación con escenarios constantes de representaciones más bajas.

Al mismo tiempo, en el contexto del *Streaming* de video, la interrupción o discontinuidad en el desplegado (situación en donde se agota el *Buffer* de video) se debe evitar como objetivo primario [14]. Este fenómeno es el más adverso de todos los que afectan a la calidad percibida de Video HTTP Streaming.

## 7.2 Métricas

La calidad de la experiencia o *QoE*, que los usuarios perciben en el desplegado de un contenido, es una medida subjetiva y cualitativa, por lo que no puede ser usada directamente, para

asegurar un mínimo en la entrega de un servicio. Por otro lado es un buen indicador de las expectativas que tienen los usuarios frente al desplegado final en un reproductor de video.

Medidas cuantitativas permiten discriminar en una escala de valores la experiencia de los usuarios frente al contenido finalmente desplegado. Con estas mediciones se pueden establecer cotas mínimas para entrega de servicio o para buscar mejoras en el desempeño del sistema de *Streaming* de video en su conjunto.

Desde el punto de vista estricto de la calidad percibida de video, es posible clasificar los indicadores de acuerdo a la forma en que son obtenidos, como: objetivos, subjetivos o mixtos.

### 7.2.1 Objetivos

De acuerdo con este criterio, se consigue el indicador basado en mediciones de atributos de la señal y un cálculo matemático directo. Por lo general se necesita la representación original de la señal de video y la final, luego de la degradación. Las relaciones simples de los valores que se obtienen dan una idea de cuanto ha cambiado.

El valor resultado de relacionar uno a uno los niveles de luminiscencia de los *Pixeles* o unidades de imagen, es un indicador del desplazamiento que ocurre desde la señal original. Comúnmente se obtiene una métrica objetiva como *PSNR (Peak Signal-to-Noise Ratio)* como valor máximo de *SNR (Signal-to-Noise ratio)* que apunta a dar una idea de la fidelidad que tiene la señal degradada frente a la original [32]. Otros esquemas de medición similares como *Video Quality Indicators, MOAVI*, se encuentran documentados en *Video Quality Experts Group (VQEG)* en enlace http://vqegstl.ugent.be/.

En las situaciones reales no se cuenta con la señal original para hacer este tipo de ratios comparativos. Para los casos *VOD* puede ser factible la aplicación de la metodología, asumiendo las demoras, mientras que para los casos *LIVE*, el video se genera en el mismo momento de consumirlo, no permitiendo la comparación, lo que limita en la práctica el uso de este mecanismo en un reproductor de *HTTP Streaming*.

El determinismo con que se consiguen los valores escalares es muy beneficioso, sin embargo no son representativos de la percepción o experiencia del usuario. Ninguno de estos resultados es directamente un indicador de la calidad percibida desplegada al usuario. El sistema de visión humano es complejo, la percepción involucra más que fenómenos instantáneos o de cuadro a cuadro, en cierta medida tiene una respuesta respecto a la evolución del contenido desplegado y a la forma en que finalmente se presenta.

Para conseguir una mejora del indicador de calidad percibida, se aplican ajustes entre los modelos QoS y QoE, como los encontrados en [31], a partir de diferencias que se registran entre la señal original y la final.

### 7.2.2 Subjetivos

En este caso se aplica una evaluación directa, por parte de los usuarios sobre una serie de videos que son preparados y desplegados, con la finalidad de obtener del grupo un *MOS* (Mean Opinion Score). Esta es una manera subjetiva de cuantificar la experiencia del participante al ver el video, o verificar el impacto de algún tipo de degradación en el mismo. Los participantes deben calificar su percepción de la calidad, asignando un valor dentro de una escala previamente explicada.

De esta forma es posible calificar la experiencia percibida del contenido desplegado, aun sin contar con la señal original de referencia. Esta modalidad se conoce como de Estímulo Simple, y es particularmente interesante esta propiedad del tipo de evaluación *MOS* en el contexto de este trabajo, ya que lo común es que no este disponible la señal original para *HTTP Video Streaming*.

De igual forma, *MOS* permite evaluar, la degradación de la señal desplegada, frente al contenido original, usando el mismo rango de valores como se muestra en Fig. 7.4. Esta modalidad se conoce como de Doble Estímulo, y se presenta en evaluación del usuario el mismo segmento de contenido, primero en su forma original y luego en su forma degradada.

| **Escala de cinco notas** | |
|---|---|
| **Calidad** | **Degradación** |
| 5 Excelente | 5 Imperceptible |
| 4 Buena | 4 Perceptible, pero no molesta |
| 3 Aceptable | 3 Ligeramente molesta |
| 2 Mediocre | 2 Molesta |
| 1 Mala | 1 Muy molesta |

Figura 7.4: Escalas de calidad y degradación de ITU-R.

Los estándares ITU-R BT.500 [37], ITU-T P.910 [30] plantean los detalles técnicos y requerimientos para la medición de la calidad percibida de video, en forma de pruebas de evaluación. Se plantean condiciones ambientales, lumínicas, formatos y distancias, así como también el tipo de prueba, de simple o doble estímulo. En cualquier caso se toma como referencia una duración de la muestra de video de 10 seg., se considera que el efecto de memoria en la percepción tiene una duración de 10 a 15 seg. y tiene un impacto exponencial decreciente en el tiempo para la calificación.

Al momento de seleccionar el contenido para la prueba se sugiere en todos los casos, usar un material neutro y representativo de los géneros disponibles. Trozos de videos donde hay mucha “acción” o con mucho “movimiento” desvían el comportamiento general de los *CODECS* usados, por los errores de codificación que agregan o la cantidad de datos que generan como representación para el contenido, frente a escenas más estables o predecibles. Por lo cual se

busca algún material neutro en estos aspectos o mezcla representativa, esta se mantiene como base para todos los escenarios a ensayar.

Se reúne al grupo de personas participantes para hacer en simultáneo el ensayo del desplegado y las encuestas, donde se presenta una secuencia de videos preparados con anterioridad. En un cuestionario los participantes colocan sus evaluaciones, con las que se calcula el *MOS*, asegurando de forma estadística un intervalo de confianza con el valor real de 95 % a partir de la desviación y el tamaño de la muestra.

El *MOS* da un indicador de calidad válido, pero tiene la dificultad de ser muy costoso en su obtención (escenario, personas, pruebas), no es posible tampoco obtener un indicador en tiempo real con fines de control dentro del reproductor.

### 7.2.3 Mixtos

Es una forma de reunir las ventajas de los enfoques anteriores, permite obtener un indicador de calidad percibida de forma rápida, automática, condiciones imprescindibles para un cuantificador en tiempo real. Además la evaluación puede conseguirse sin tener la señal original de referencia, solo tomando como datos de entrada valores objetivos en las mediciones indirectas disponibles en forma local al sistema de reproductor de contenidos.

#### 7.2.3.1 PSQA

Se muestra como ejemplo la metodología *PSQA (Pseudo Subjective Quality Assessment)* que obtiene el indiciador evaluando una función especial de PSQA ajustada por entrenamiento. La selección de los parámetros, así como el ajuste de la función es parte del mecanismo de la metodología [5].

Pasos para obtener una función PSQA como indicador:

- Determinar un conjunto de parámetros de impacto en la calidad de video.
- Construir un escenario de pruebas con el que sea posible ver secuencias de video distorsionadas en forma controlada según los parámetros seleccionados.
- Configurar un ensayo y un panel humano para realizar una encuesta subjetiva de las secuencias.
- Eliminar muestras malas, que son aquellas que difieren significativamente de la opinión media.

- Calcular el *MOS* en base a las muestras. Estos valores determinan para cada configuración distorsionada, la medida de calidad.

- Entrenar una red neuronal, usando como entrada los parámetros relevantes, para obtener como salida un índice, que debe ser un estimador del *MOS* ensayado.

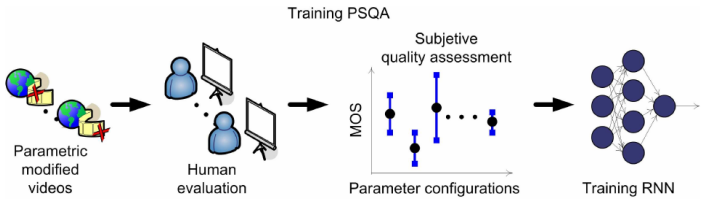


Figura 7.5: Esquema de inferencia *PSQA* [5].

Cuando la red neuronal está entrenada, permite reconocer que tipo de evolución ocurre en el sistema y devolver el valor *MOS* asociado a partir de sus parámetros indirectos que caracterizan al desplegado.

#### 7.2.3.2 Reconocimiento de Patrones

Otra propuesta factible es reconocer patrones en la evolución del contenido desplegado, por ejemplo el esquema presentado en [10] dentro de otros pre-calificados. Las variables registradas por sí solas durante el desplegado de contenido, son una referencia indirecta del desempeño final de la calidad percibida.

En cambio la secuencia multi-dimensional de las mismas, permite conformar patrones de evolución. Estos patrones permiten ser clasificados según la evaluación de calidad percibida que obtuvo, con un método de aprendizaje automático, por ejemplo *KNN: K-Nearest Neighbor* 8.1. El proceso funciona en dos fases: una de entrenamiento y otra de evaluación.

En la etapa de entrenamiento, se despliegan los contenidos que se califican y etiquetan con *MOS* evaluado, mientras se registra la secuencia de valores en las variables objetivas de interés. De esta forma se genera un conjunto de referencia que se incorpora al sistema y lo habilita para la segunda fase.

En la segunda fase el sistema es capaz de evaluar un contenido nuevo y dar el *simil-MOS*, a partir de la secuencia de indicadores indirectos. El mecanismo consiste en encontrar dentro del conjunto de referencia, los $K$ elementos más cercanos e inferir de estos el indicador, usando el promedio de sus etiquetas.

La cercanía de los patrones involucra la noción de distancia, en este problema, la clasificación es dependiente de la forma en que se calcula, en adelante se considera como ajuste del propio método. Se consiguen buenos resultados con el uso de la distancia euclidiana o con otros criterios relativos a la comparación de la forma que tienen los valores de la secuencia.

En el trabajo se usa con buenos resultados, la distancia *DTW: Dynamic Time Wrap* entre secuencias, que permite reconocer la ”forma” de la secuencia, cuando se flexibiliza el eje temporal, se explica en [26].

*DTW* parece funcionar bien para videos del orden de algunos minutos, tomando valores promedios para las muestreas en intervalos de 10 a 15 segundos.

Se consideran como atributos relevantes para conformar los registro de secuencia el valor de *kbps Bandwidth* disponible para la descarga, cantidad de paquetes recibidos, *pkts/seg*, cantidad de *re-Bufferings* y *Frame-rate fps*.

En la Figura 7.6 se muestra el proceso en las etapas de entrenamiento *(Training)* y de evaluación *(Rating)* [10].

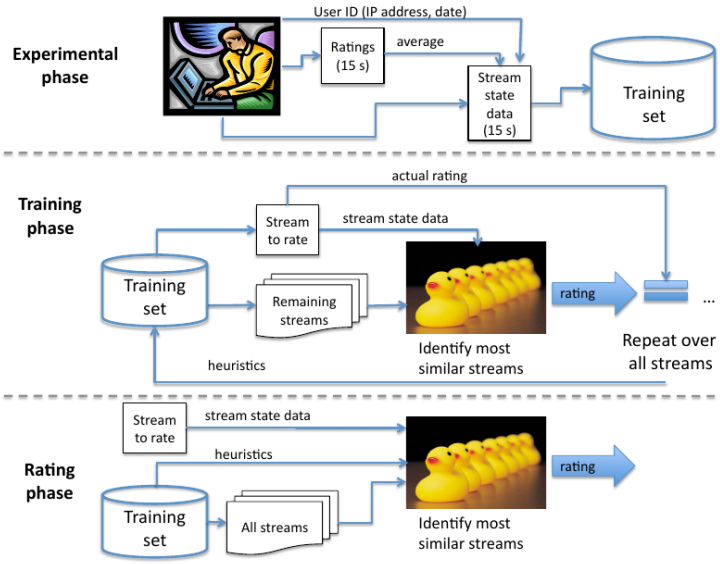


Figura 7.6: Esquema de inferencia por reconocimiento de patrones [10].

Este método de evaluación para obtener *MOS*, implica la formación de patrones para los cuales se eligen las variables de más impacto en el desplegado final. La dimensión máxima de la secuencia depende del método usado para medir la distancia, teniendo en cuenta que el cálculo de DTW no es simple, el autor se limita al uso de algunas variables.

# Capítulo 8

# Uso de KNN como medida de calidad de Video

A continuación se presentan resultados de casos de estudio similares para la cuantificación de la calidad percibida. Es interesante observar estos resultados, para ver cuales son los factores que afectan. En particular se hace foco en este trabajo en los que están disponibles y que son medibles al nivel de abstracción de la heurística. El mismo nivel de abstracción donde es posible tomar opciones de selección con influencia en la experiencia del desplegado.

En general se tiende a esquemas que optimizan las transferencias de los segmentos de video en función de los recursos como la disponibilidad de ancho de banda, restricto a condiciones mínimas, exigidas para el mantenimiento de *Buffer* con holgura.

Se asume que una mejor experiencia para el usuario, está asociada a la reproducción de representaciones más fieles al contenido original que se pueda desplegar en el reproductor.

En estos casos se aplica algún tipo de modelo que relaciona parámetros de QoS para las transferencias, con la continuidad del desplegado. Sin embargo según está planteado en ITU-T P.910 [30]. la percepción involucra otros fenómenos además de los instantáneos, y queda condicionada temporalmente por la evolución del desplegado final.

Existe una memoria perceptiva que influye en el cuantificador de calidad percibida. Estos fenómenos deben ser tomados en cuenta por los sistemas de evaluación a partir de medidas objetivas indirectas. Esta memoria perceptiva es del orden de 10 a 15 segundos y su influencia tiene un perfil de tipo exponencial decreciente hacia los tiempos más antiguos [30].

En este trabajo se plantea el reconocimiento y clasificación de escenarios similares, en forma de patrones, a partir de variables objetivas, similar a como es utilizado en [10] para videos cortos del orden de minutos. Se utiliza *K-Nearest Neighbor* como mecanismo de reconocimiento para patrones. En el estudio de referencia, los resultados conseguidos como un indicador equivalente a *MOS*, no tienen diferencia con el valor real en el 75 % de tiempo.

## 8.1 Nearest Neighbor

*K-Nearest Neighbor* [40], conocido como *KNN*, es un algoritmo de aprendizaje automático para problemas de clasificación, que genera una estructura de almacenamiento para el conocimiento que recibe a partir de datos de un ambiente, donde en forma supervisada se asocia un concepto para cada elemento.

Los algoritmos supervisados intentan extraer aquellas propiedades que permiten discriminar mejor la clase de cada elemento o patrón y como consecuencia requieren de una clasificación previa (supervisión) del conjunto de entrenamiento.

El aprendizaje se define como el proceso mediante el cual, un sistema mejora y adquiere destreza en la ejecución de su tarea. Tiene la capacidad de inferencia inductiva a partir del conocimiento que toma del ambiente [6]. Un modelo de aprendizaje puede caracterizarse por dos cuerpos de información: ambiente y base de conocimiento, y por dos procedimientos: elemento de aprendizaje y elemento de ejecución, tal y como se representa en la Figura 8.1. En este caso, los elementos que forman el conjunto de entrenamiento normalmente se componen por pares del estilo [objeto de entrada, clase del objeto], donde el objeto de entrada suele estar representado por un vector de atributos (o propiedades del objeto). La misión de los algoritmos de clasificación de aprendizaje automático supervisados, es encontrar el conjunto de atributos que permite predecir con mayor precisión la clase de cada objeto del conjunto de entrenamiento.

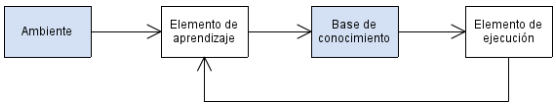


Figura 8.1: Elementos del modelo de aprendizaje.

En el proceso de aprendizaje, no se hace ninguna suposición sobre el tipo de distribución de probabilidad de las variables. Esta propiedad lo hace muy interesante para caracterizar/clasificar datos experimentales.

Las reglas de clasificación están basadas en la búsqueda dentro del conjunto de elementos prototipos cercanos al patrón a clasificar. La noción de distancia euclidiana, permite evaluar la semejanza entre los elementos. El método en si mismo no plantea el uso de una norma específica para evaluar la distancia, en [2] se pueden ver otras alternativas.

*KNN* se considera un algoritmo de tipo *Lazy Learning* (perezoso) en la fase de aprendizaje, el costo es nulo, todo el cómputo es diferido al momento de la clasificación.

**Algorithm 1** Algoritmo KNN(Q,R,K) que retorna los índices en R de los K elementos más cercanos a Q

**Require:** Elemento de consulta $Q = (1 \times \dim_M)$ no vacío.
**Require:** Base de conocimiento $R = (N \times \dim_M)$ no vacío.
**Require:** $K \geq 1$
{Inicialización}
$dim = columnas\,(Q)$
$largo = filas\,(R)$
$ind = ceros\,(K)$
$distancias = ceros\,(largo)$
{Cálculo de distancias para cada fila por columnas}
**for** $t = 1; t \leq dim; t++$ **do**
$distancias = distancias + ((R\,(:,t) - Q\,(1,t)))\,.^2$
**end for**
{Ordenar ascendente distancias}
$ind = Odenar(distancias)$
{Seleccionar los primeros k elementos}
$ind = ind[1..k]$
**return** $ind$

El procedimiento de búsqueda sigue el Algoritmo 1, ordena en forma ascendente por distancia, los elementos encontrados próximos al nuevo a clasificar *Q*, del conjunto de prototipos o base de conocimiento *R*.

Como resultado se puede inferir una propiedad desconocida de un nuevo elemento, a partir de la base de conocimiento y los K-elementos resultado más próximos. Para obtener el valor final, algunas de las posibilidades para procesar las etiquetas de los elementos resultado son: el promedio, el promedio ponderado por la distancia, o la mayoría.

### 8.1.1 Hipótesis de KNN

Para utilizar este tipo de minería de datos se deben cumplir algunos requerimientos:

- Debe ser posible establecer una métrica de distancia entre vectores patrones que caracterizan a los elementos a clasificar.

- Debe existir una vecindad en el dominio del problema, en donde los elementos de la misma clase, deben formar grupos o regiones contiguas (en la noción de distancia usada) y estas no deben superponerse.

Se muestra algunas definiciones de distancia en [2] como: *Euclidiana, HVDM, HEOM, VDM, normalizada*, dependiendo del tipo de problema algunas normas funcionan mejor que otras. Así como se recomienda una normalización de las variables para el uso de distancia euclidiana, para evitar mayor influencia por parte de los componentes con valores absolutos mayores.

Si bien no es necesario un ajuste del método, si se necesita caracterizar los diferentes escenarios para cada problema en particular. En la fase de aprendizaje o entrenamiento, las variables disponibles dentro de los bloques del reproductor (Figura 5.1), como por ejemplo las que aparecen en la Figura 6.1, deben conformar un vector patrón que sea hábil para la clasificación que se busca.

Se usará *KNN* como método regresivo, que da un valor a partir de los etiquetados de patrones que caracterizan el comportamiento del sistema de desplegado de video. Escenarios equivalentes en este contexto, son representados por vectores próximos en este modelo, y por lo tanto tienen el mismo nivel de calidad percibida.

Existe un compromiso para determinar el valor apropiado de $K$, que afecta tanto a la robustez del método frente al ruido, como a la precisión del valor inferido.

En el dominio de los vectores patrones, pueden existir regiones con diferente densidad de puntos, donde un conjunto de $K$ elementos no distinga con igual claridad a cada clase. En este trabajo se busca un valor apropiado para obtener un estimador suficientemente bueno, en función del error medio al valor real de la inferencia.

### 8.1.2 Validación

Si para validar las predicciones de clase del método, se utilizan las muestras usadas en la formación del aprendizaje, la exactitud que se obtendrá en el valor inferido será artificialmente mayor que el real [23]. Se prefiere que con fines de validación se usen conjuntos de datos disjuntos a los usados en la etapa de entrenamiento.

Se evalúa típicamente el rendimiento de los clasificadores a través de la validación cruzada, la precisión de clasificación mejora, con múltiples particiones de las muestras de entrada utilizadas en el aprendizaje. Este mecanismo se llama *k-Fold* [1], en donde una de las particiones que no participa en el entrenamiento, es usada para la validación. Como criterio de validación se usa el error medio cuadrático del valor inferido con el real.

[1] https://en.wikipedia.org/wiki/Cross-validation_(statistics)

## 8.2 Normalización

Con la finalidad de incrementar la capacidad del método, se hace una transformación de los valores de entrada, previo a las etapas de aprendizaje. Los procesamientos de las variables que dan buenos resultados según [23] son: *Standardization* y *Fuzzification*.

La estandarización remueve los efectos causados por problemas de escalado entre los valores de variables. Estas, en distintos ordenes de magnitud, pueden influenciar o desplazar al clasificador. La transformación que se aplica a la medida cruda es de tipo *Z-score*, donde se aplica la información que aporta la media y la desviación estándar de las muestras.

$$\mu(OS) = \widetilde{OS}$$

$$s(OS) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(SO_i - \mu(OS))^2}$$

$$\text{z-score}_n = \frac{OS_n - \mu(OS)}{s(OS)}$$

En la *Fuzzification* la transformación propuesta aprovecha la incertidumbre en los valores para incrementar el rendimiento del clasificador. En este proceso se aplica un mapeo a un nuevo valor, característico del conjunto al cual pertenece el valor original [22]. Los conjuntos quedan determinados por los rangos de valores originales en forma de cuartiles donde cada uno tiene un valor de retorno.

En la fase de aprendizaje, durante la caracterización de los escenarios que se presentan a los usuarios, se asocia una etiqueta representativa de la calidad percibida por el grupo. Cada observador califican su percepción optando por un valor discreto en un escala de 1 a 5. Las encuestas de los observadores se deben compensar, para descartar sesgos como: el efecto de su experiencia personal al evaluar, las preferencias por el tipo de contenido, particularidades de cada observador y rangos usados al calificar.

La aplicación de una normalización en la calificaciones, con el criterio de *Z-score* según [23] da mejores resultados en el uso de *KNN*. Con las evaluaciones normalizadas para cada muestra, se calcula el correspondiente *MOS*, como el valor promedio entre los usuarios. El *Z-score* es una variable comparable entre escenarios diferentes, que permite un mejor etiquetado de los patrones con la finalidad de mejorar la eficiencia del clasificador.

## 8.3 Resumen

Como se ha presentado, conseguir una métrica comparable a *MOS* a partir de inferencias en variables indirectas que afectan al sistema, tiene sus dificultades. La exactitud con el valor real, no es total y el costo del cálculo puede ser un problema en algunos casos.

Para continuar con este trabajo en el Capítulo 9 se busca conseguir un indicador suficientemente bueno para estimar *MOS* basado en *KNN*. En la última parte se plantea un mecanismo de estimación eficaz, que se validará con criterio de error medio entre el valor inferido por el método y el valor real.

# Parte IV

# ENSAYOS Y CONCLUSIONES

# Capítulo 9

# Ensayo

## 9.1 Estimación de calidad percibida en HAS

Como primer objetivo se plantea el experimento para obtener un mecanismo equivalente a la evaluación *MOS*, a partir de los datos del ensayo. Para lograrlo, se utiliza el mecanismo de minería de datos revisado en la Sección 8.1. En el análisis de resultados se muestra su efectividad.

## 9.2 Impacto de la heurística en la Calidad percibida

Luego de tener disponible un mecanismo para evaluar la calidad percibida, no es necesario obtener el *MOS* a partir de calificaciones de usuarios. Esto permite repetir los ensayos, con un costo menor y obtener un indicador equivalente a *MOS* de una manera más simple. Es posible entonces alcanzar un segundo objetivo, y teniendo en cuenta que los escenarios usados son deterministas, se pueden hacer ensayos comparables entre heurísticas.

Se utiliza entonces el indicador para valorar el resultado promedio de algunas de las heurísticas que aparecen en el Cuadro 9.1.

| Nombre | Descripción |
|---|---|
| JWPlayer original | Heurística descripta para *JWPlayer* en Sección 6.3 |
| JWPlayer positivo | Variante de heurística JWPlayer, sin restricciones para cambios ascendentes (Fig. 6.6) |
| Fijo | Heurística sin dinámica: valores fijos y predeterminados para el tipo de segmento a descargar |

Cuadro 9.1: Heurísticas de ejemplo para las pruebas.

## 9.3 Planteo del experimento

En la Sección 7.1 se muestran los factores de mayor incidencia en la experiencia desde el punto de vista del reproductor, esto se resume en el Cuadro 9.2:

| |
|---|
| El tiempo de inicio de reproducción |
| La cantidad de *Re-Buffering* |
| La duración en cada instancia de detención |
| El tipo de representación de la fuente de video |

Cuadro 9.2: Factores que afectan a QoE.

La calidad de servicio que la infraestructura de acceso presenta al reproductor, queda determinada principalmente por los factores mostrados en el Cuadro 9.3:

| |
|---|
| El *Bandwidth* o ancho de banda efectivo del cliente |
| El *Delay* o demoras entre el origen del contenido y el reproductor |

Cuadro 9.3: Factores que afectan a *QoS*.

En *HTTP* las pérdidas de información en el transporte de datos, se transforman en demoras y un menor ancho de banda efectivo, como se analiza en la Sección 5.2. En situaciones reales, el reproductor está fuertemente condicionado en su funcionamiento por los factores de red, siendo estos últimos los estudiados en este capítulo. La variabilidad genera cambios en la disponibilidad de acceso a los datos de origen, y no permite mantener un desplegado uniforme en los sistemas de *HTTP Streaming*.

Con la finalidad de evaluar la calidad de la experiencia, se plantea un ensayo en donde se presentan escenarios diversos, de calidad de servicio, a un reproductor de video. Los escenarios que se imponen, impactan de forma indirecta en la calidad percibida.

En el resto del capítulo se muestra como es el comportamiento de los factores que afectan a QoE enumerados en el Cuadro 9.2 frente a la influencia de las perturbaciones externas. Estos factores son medibles y se registran como evaluación de la evolución del sistema.

Se propone obtener un indicador similar a *MOS*, a partir de variables registradas en el mismo nivel de abstracción en donde la heurística opera. Los segmentos de video son tratados en forma atómica o indivisible en el contexto de las tecnología HAS y es de interés en este nivel de abstracción la información disponible acerca de su accesibilidad.

La ejecución de test subjetivos para estimar el MOS es costosa y solo se realiza una vez. Las variables relevantes aparecen a lo largo de la trayectoria que sigue el flujo de los datos. El comportamiento dentro del reproductor de video queda registrado en muestras tomadas ocho

veces por segundo para las variables de interés y en simultaneo la calidad percibida por cada usuario.

Terminado el ensayo se busca un indicador comparable a *MOS*, mediante un mecanismo de inferencia, basado en un modelo de aprendizaje/clasificación mostrado en el Capítulo 8. Durante el proceso de aprendizaje, se compone un vector patrón a partir de los registros obtenidos, que sea útil para la clasificación de los escenarios en función de su evaluación *MOS*.

A cada registro original, se le agregan copias de otros anteriores, que se encuentren a un número entero en unidades de desplazamiento del actual. La unidad de desplazamiento usada fue de 2 segundos y el máximo desplazamiento aplicado para copiar un registro anterior fue de 20 segundos. Este mecanismo se propone para evaluar la existencia de correlaciones con variables que se encuentren desfasadas en tiempo.

Los patrones que finalmente se ejercitaron fueron promedios en algunas componentes seleccionadas del registro ampliado.

Con la finalidad de quitar efectos "estacionales" el patrón que caracteriza al escenario, se calcula con promedio para cada una de las componentes, en un número consecutivo de registros. Este enfoque refleja con buena precisión los cambios de nivel y sentido en las variables de interés en un espacio temporal comparable al largo de un segmento.

Tanto el proceso de selección como la elección del número de registros consecutivos, son ajustes del método de inferencia y se detallan en Capítulo 10.

## 9.4 Plataforma de pruebas

En la Figura 9.1 se muestra un diagrama con los elementos para hacer la prueba y la obtención de datos. Se imponen condiciones variables en el acceso a los segmentos de video con el uso de un emulador de red entre el origen y el reproductor de video.

Se requiere emular el acceso a los segmentos de origen, aplicando una diversidad de escenarios, que provoquen dentro del reproductor, diferentes puntos de funcionamiento. El mecanismo de inferencia dará mejores resultados, si en su entrenamiento, la diversidad de los escenarios presentados es mayor y cubre todo el dominio de posibilidades.

Estos escenarios deben ser equivalentes a las condiciones que aparecen en el funcionamiento real, lo que se consigue aplicando combinaciones (*Bandwidth, Delay*) similares a las que se encuentran en la práctica, como se ve en la Figura 9.2.

En la Figura 9.3 se muestra los valores que se obtienen de forma simultánea a las evaluaciones de calidad de los usuarios. Los registros guardados en la prueba consolidan la información

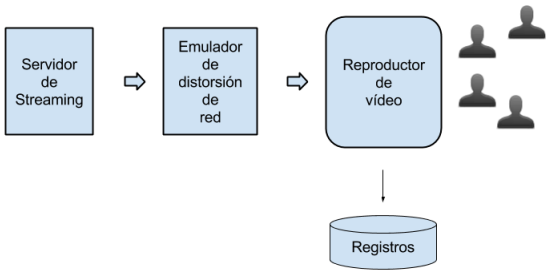


Figura 9.1: Elementos del ensayo.

```
tc class change dev eth0 parent 1: classid 1:1 htb rate 0.5mbit
tc qdisc change dev eth0 parent 1:1 handle 10: netem loss 0.5% delay 100ms 20ms
```

Figura 9.2: Tipo de reglas aplicadas al emulador de red.

en forma de muestras a una tasa de 8 veces por segundo. La realización de una ensayo tipo *MOS* es costoso, y es preferible luego de terminada la prueba, buscar cuales pueden ser los mejores vectores patrones proyectando los registros almacenados.

En el Cuadro 9.4 se resumen las variables que se registran como datos del ensayo:

Al momento del registro, los valores representan el estado de las variables en un instante del tiempo. Se necesita información sobre el comportamiento en la evolución de estas variables que caracterizan al escenario. Se aplican desplazamientos del orden de segundos, sobre registros anteriores, agregando dimensiones al presente.

Este registro, ampliado, de mayor dimensión contiene la evolución temporal. En este trabajo se compone un patrón sencillo consolidando cada dimensión de la secuencia en forma de promedios.

El uso de diferentes proyecciones de los datos presentes en el registro ampliado, permite cuantificar con el criterio de validación e identificar cuales tipos de vectores patrones funcionan mejor como clasificadores. El *MSE* error medio cuadrático, es un criterio apropiado para validar la precisión del identificador estimado.

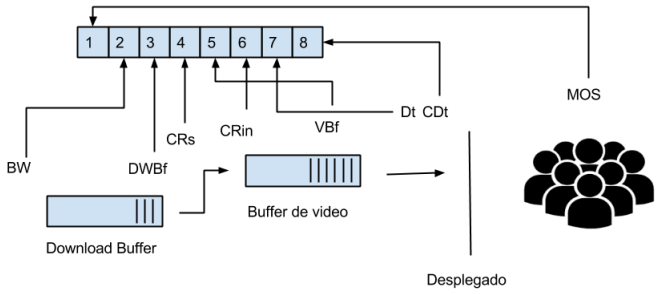


Figura 9.3: Valores de parámetros que afectan indirectamente QoE

| | |
|---|---|
| 1 | Evaluación de calidad del usuario |
| 2 | *Bandwidth (BW)* |
| 3 | Largo de *Download Buffer (DWBf)* |
| 4 | Próximo *Code Rate* seleccionado (CRs) |
| 5 | Largo de *Buffer* de video *(VBf)* |
| 6 | *Code Rate* que ingresa al *Buffer* de video *(CRin)* |
| 7 | Detención de desplegado *(Dt)* |
| 8 | Cantidad de detenciones *(CDt)* |

Cuadro 9.4: Factores registrados en el ensayo.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(\widehat{MOS_i} - MOS_i\right)^2$$

Donde $MOS_i$ es el resultado del ensayo y $\widehat{MOS_i}$ es el estimador.

En este caso de estudio, las variables naturalmente tienen rangos y valores similares, por lo que se mantienen los originales, sin uso de procesos de escalado.

Durante el proceso de aprendizaje mostrado en 8.1 se verifica la proximidad de los elementos de una misma clase, con el porcentaje de superposición, (medido como cantidad de elementos de menor distancia a la mitad de la media entre diferentes clases). Se encuentra como razonable no superar el 5 % de superposición.

Finalmente queda determinar el valor de K para el algoritmo, este valor hace referencia a la cantidad de elementos cercanos al de consulta usados en la inferencia. Los valores sugeridos en las revisiones de [2], están entre 5 y 11, según el caso de estudio o el problema. Se opta por un valor de K=7 que da buenos resultados para los datos del ensayo.

## 9.5 Preparación del experimento

La tecnología *Adaptive Streaming* para video, utiliza segmentos independientes y codificados con diferentes parámetros en su origen, principalmente *Frame-Rate*, *Code-Rate*, manteniéndose el ratio ancho/alto. Los segmentos que se generan son de diferentes tamaños dependiendo del contenido a desplegar o información que contienen en función de la exactitud de su representación.

Para obtener resultados iniciales en un experimento de tipo MOS, el estándar ITU-T P.910 [30] sugiere una duración estandarizada de 10 segundos para cada contenido a desplegar. Además fija también un mínimo de unas 200 muestras para realizar el ensayo, equivalentes a unos 35 minutos de contenido continuo. Asumiendo que cada segmento de video es una prueba independiente, la cantidad de datos relevados es equivalentes a una prueba estándar de video.

El reproductor usado en este caso, consume segmentos de aproximadamente ocho segundos de duración en protocolo *HLS*. Un total de 38 segmentos componen el contenido a desplegar, equivalentes a una duración de 5 minutos en tiempo de contenido. Con una presentación visual variada. El experimento consiste en la aplicación secuencial de los escenarios del ensayo durante la visualización de todo el contenido. Realizando la prueba en seis ocasiones, se consigue un conjunto de datos válidos.

### 9.5.1 Representaciones del contenido

En este ensayo se usa como fuente de contenido un solo video, llamado “EXTRA”, en el Cuadro 9.5, se muestran las opciones de codificación que describe el manifiesto:

Audio: HE-AAC or AAC-LC, Stereo, 22.05 kHz, ab=40kbps
Video: EXTRA, Ratio ancho/alto=16:9, Baseline profile=3.1

| REP (Q) | RESOL | kbps | KeyFrame |
|---|---|---|---|
| 9 | 1280x720 | 4540 | 90 |
| 8 | 1280x720 | 2540 | 90 |
| 7 | 960x540 | 1800 | 90 |
| 6 | 960x360 | 1200 | 90 |
| 5 | 960x360 | 600 | 90 |
| 4 | 480x224 | 400 | 90 |
| 3 | 960x224 | 200 | 45 |
| 2 | 256x144 | 110 | 30 |
| 1 | 128x72 | 64 | 15 |

Cuadro 9.5: Tipos de representación para cada segmento de video

En este trabajo se considera un tipo de representación de segmento de video como una resolución y cuantización dada, estos configuraciones son recomenadaciones de Apple para el protocolo *HLS* [1].

### 9.5.2 Escenarios

La prueba se ejecuta con perfiles de restricción sobre el ancho de banda efectivo que encuentra el reproductor para descargar los segmentos de video desde el origen. Estos son similares a los documentados en [1] para ensayos parecidos.

Las variaciones impuestas en la emulación de red hacen que el reproductor de video vea modificadas las condiciones para acceder a los segmentos, aplicando de igual forma en todos los escenarios una demora de 100 mseg +-20, con pérdida de paquetes ajustado al 0.5 %. Las condiciones quedan caracterizadas en el tiempo, por tipos de perfiles para el ancho de banda. En la Figura 9.4 se muestran en detalle cada uno de ellos: A, B, C, D. Estos tienen una tasa ilimitada de *5 mbps* (suficiente para operar el reproductor de video en la representación superior) con restricciones por períodos de *0.5mbps* (con el fin de disminuir la capacidad de la red, sin cortar el flujo de datos).

El ensayo total tiene una duración aproximada a los 30 minutos, por lo que se deben repetir estos perfiles, para la extensión del contenido. Se elige una mezcla de forma: D-A-B-C-B-A-D-C solo con la idea de poder repetir el experimento con escenarios similares, podría haberse usado cualquier otra combinatoria. El objetivo de los perfiles es cubrir con suficiente diversidad los escenarios de perturbaciones y el momento en que aparecen. El *Bandwidth* efectivo promedio para el ensayo queda en 1.83 mbps.

[1] https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html

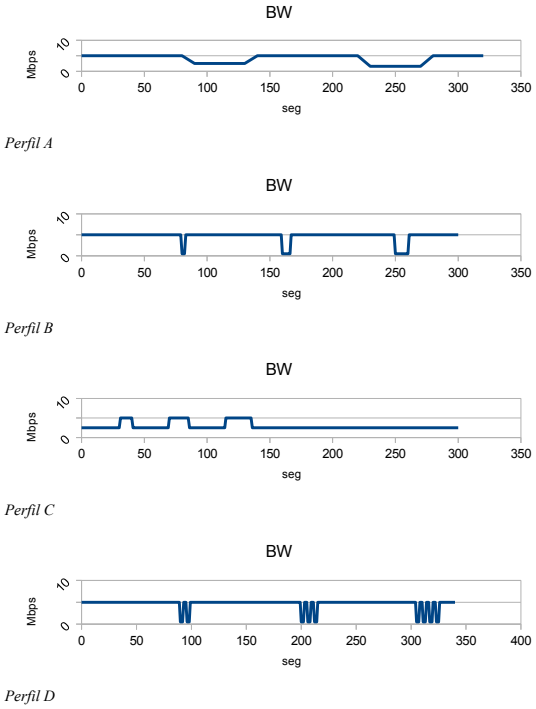


Figura 9.4: Tipos de perfiles que se imponen al sistema para generar variabilidad.

# Capítulo 10

# Resultados

## 10.1 Estimación de calidad percibida en HAS

En este capítulo se presentan en resumen los datos obtenidos como resultado del ensayo, y como a partir de estos se obtiene una valoración estimada de *MOS*.

El uso de regresión lineal múltiple [38], permite encontrar cuales son los componentes más apropiados para construir un vector patrón representativo a utilizar en la etapa de aprendizaje del método de inferencia. Como se propuso en la Sección 9.4, a partir de las variables registradas, se construye vector de dimensiones ampliadas con el agregado de columnas con las mismas variables y en el mismo orden, desfasando el conjunto original en unidades de tiempo de 2 segundos. Los componentes relevantes de este vector ampliado se proyectan para formar el patrón y que se usará en la clasificación de escenarios.

$$MOS = \overrightarrow{b} \times Vec_{pat}$$

El resultado de $\overrightarrow{b}$ muestra en magnitud una aproximación de la influencia relativa entre los componentes.

Se verifica como se menciona en ITU-T P.910 [30], un perfil descendente para las influencias de los componentes del registro ampliado sobre MOS. En la medida que se alejan en el tiempo del momento de evaluación, el efecto se atenúa como se ve en la Figura 10.1, donde se observa como cambia la correlación a medida que los grupos de componentes originales son desplazados en el tiempo, desde el momento de evaluación.

En algunos componentes la influencia no es inmediata, sino que hay un desplazamiento temporal hasta su máxima correlación. Este corrimiento temporal es del orden de segundos, entre la medición registrada y el momento de la evaluación del usuario. Esto es equivalente a decir que el mayor impacto de los factores objetivos medidos ocurre, cuando son registrados

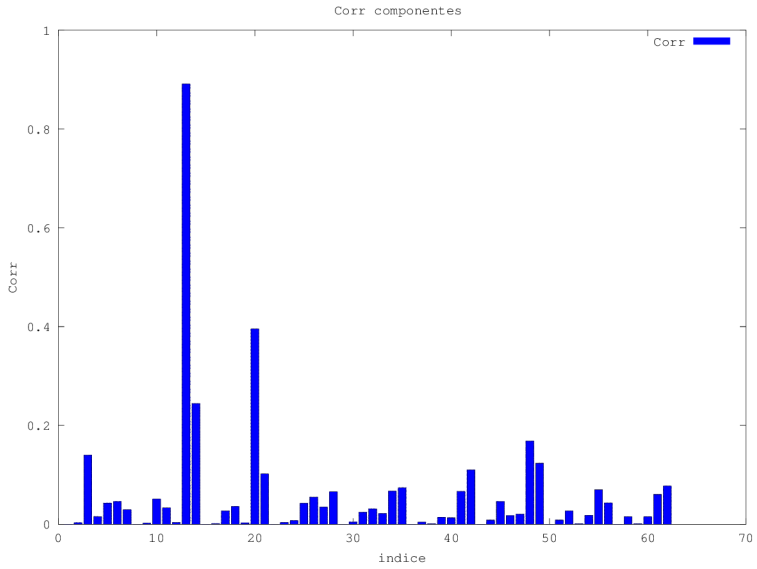


Figura 10.1: Impacto relativo de las variables en la calidad percibida para el registro ampliado para cada componente.

más "cerca" del observador [30].

En las Figuras: 10.2, 10.3, 10.4, 10.5, se muestran cuales de las componentes del Cuadro 9.4 son mas influyentes en el MOS. Las gráficas muestran para cada una que correlación tienen con la calidad percibida, en función de su desplazamiento temporal, en unidades de 2 segundos. El desplegado para el observador, tiene mejor correlación con factores internos próximos al usuario o de mayor nivel de abstracción. Esta característica es de interés en el contexto de *Adaptive Streaming HTTP*, porque relaciona fuertemente la disponibilidad de los segmentos y su tipo de representación con la calidad percibida.

Como resultado se obtiene una valuación similar a MOS para *Adaptive Streaming HTTP* independiente de las medidas tomadas para los anchos de banda efectivos y volumen de datos a descargar. Esta propiedad del estimador MOS conseguido, lo hace independiente de los bloques o capas subyacentes que proveen segmentos a las etapas finales del desplegado.

Este proceso ya fue usado por [10] para etiquetar videos cortos, de 1 a 2 minutos con muy buena aproximación. En este caso de estudio, el lapso de tiempo en el que se desarrolla la evaluación es mucho menor, del orden del largo de un segmento.

La elección de componentes está basada en la influencia relativa encontrada por regresión, según $\overrightarrow{b}$. Finalmente solo algunas de las variables registradas, son efectivas en el funcionamiento del método de inferencia. En el Cuadro 10.1 se describen las componentes del vector que se toma como patrón clasificador.

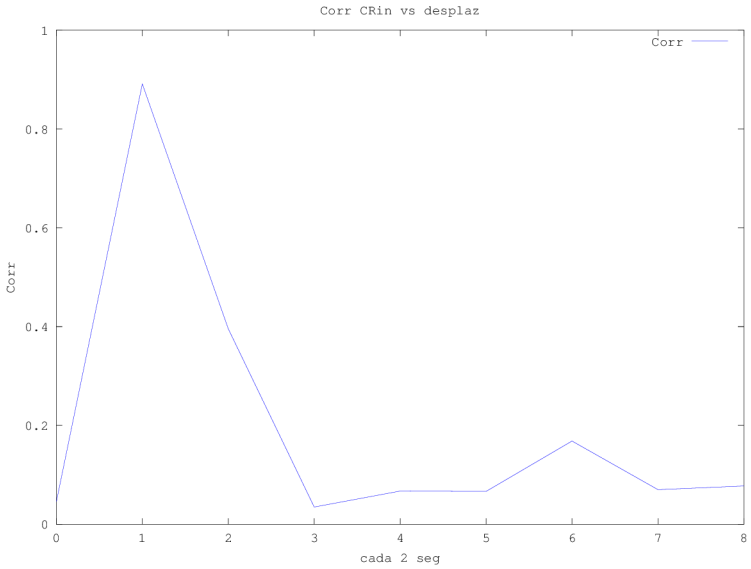


Figura 10.2: Relevancia del *Code Rate* que ingresa al *Buffer* de video.

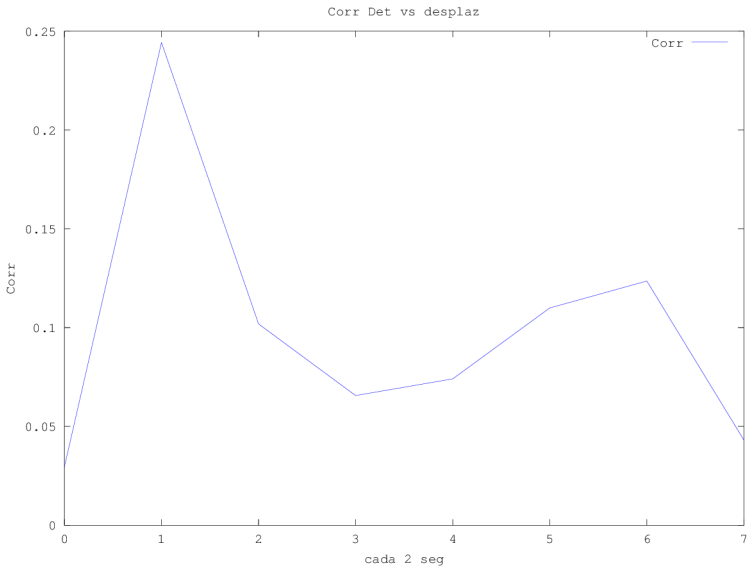


Figura 10.3: Relevancia de la detención del desplegado.

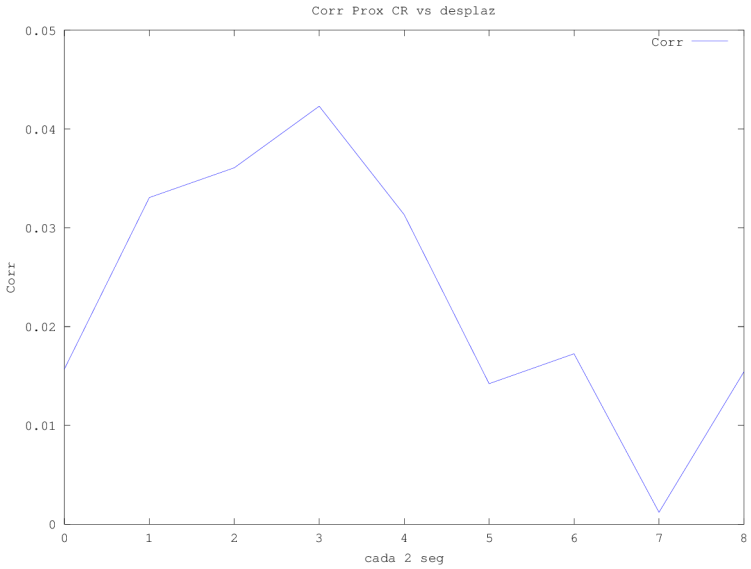


Figura 10.4: Relevancia del próximo *Code Rate* seleccionado.

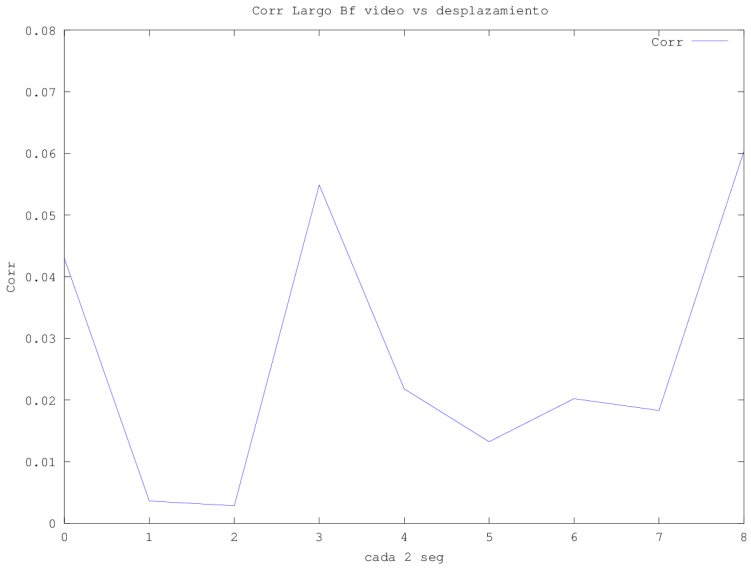


Figura 10.5: Relevancia del tiempo restante de *Buffer* de video.

| Índice | Registro Original | Desplazamiento en seg. |
|---|---|---|
| 26 | Largo de *Buffer* de video *VBf* | 6 |
| 13 | *Code Rate* que ingresa al *Buffer* de video *CRin* | 2 |
| 14 | Detención de desplegado *Dt* | 2 |
| 25 | Próximo *Code Rate* seleccionado *CRs* | 6 |

Cuadro 10.1: Vector patrón final utilizado por *KNN*.

Con la finalidad de verificar la precisión del estimador MOS conseguido se hace un estudio de *MSE*, error medio entre los valores inferidos y los reales. En la validación el valor *MSE* se obtiene como promedio de los resultados de aplicar un *K-Folding* en las etapas del entrenamiento, con valor k=5, permitiendo comparar el valor MOS real con el inferido (ver Figura 10.6).

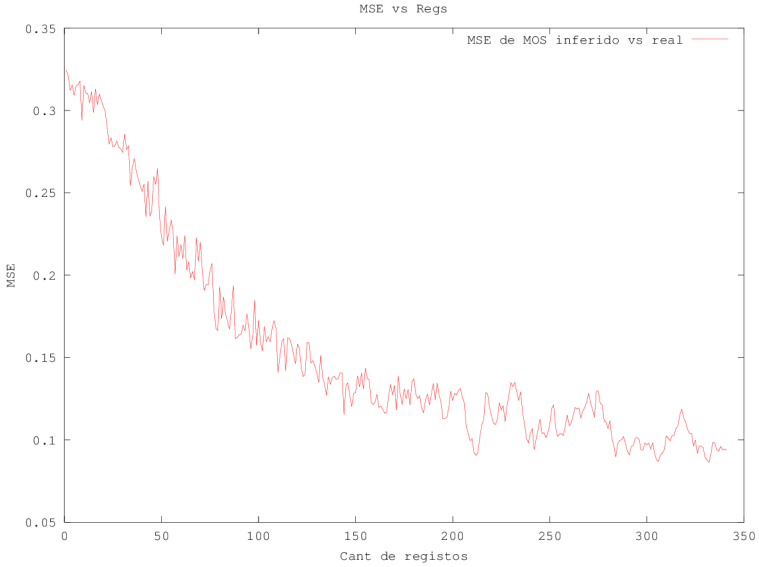


Figura 10.6: Error medio del valor estimado $\widehat{MOS}$ con el $MOS$ vs el número de muestras en promedio.

El *MSE* permite dirigir la fase de entrenamiento de *KNN*, tomando las proyecciones del registro ampliado mostrado antes, que favorecen el mecanismo de inferencia. Al mismo tiempo también es útil, en conjunto con los resultados de la Figura 10.7, para determinar cual es la cantidad de registros consecutivos más conveniente a tomar para calcular un valor promedio.

Los resultados validan el método para inferir $\widehat{MOS}$ a partir de una secuencia de registros de variables objetivas indirectas. De acuerdo al Cuadro 10.2 el error medio del estimador $\widehat{MOS}$ es menor que cualquiera de los participantes del ensayo. Con este resultado se considera que el estimador es un mecanismo hábil para la estimación de MOS.

## 10.2 Impacto de la Heurística en la Calidad percibida

El mecanismo de inferencia propuesto en la sección anterior permite obtener resultados reales comparables, como se ve en la Figura 10.8.

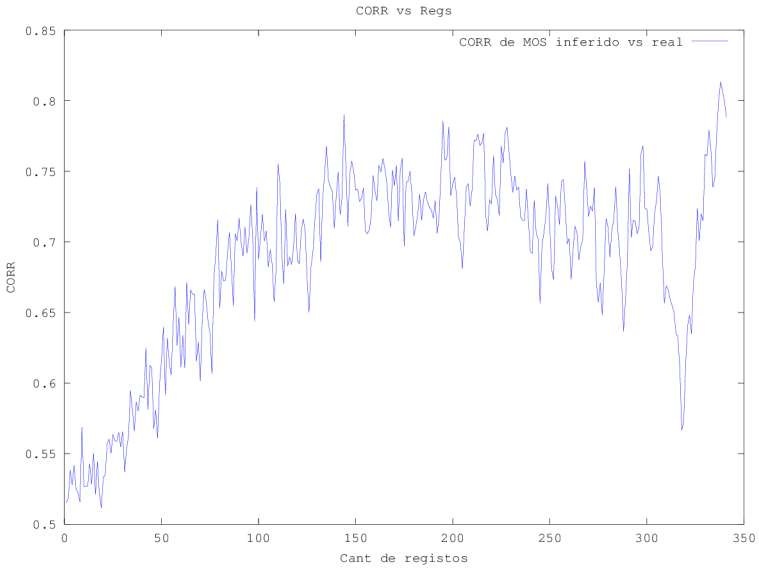


Figura 10.7: Correlación del valor estimado $\widehat{MOS}$ con el $MOS$ vs el número de muestras en promedio.

| Individuo | MSE a MOS | RMSE a MOS |
|---|---|---|
| 1 | 0.45 | 0.67 |
| 2 | 0.74 | 0.86 |
| 3 | 0.44 | 0.66 |
| 4 | 0.30 | 0.55 |
| 5 | 0.52 | 0.72 |
| 6 | 0.75 | 0.87 |
| 7 | 0.64 | 0.8 |
| $\widehat{MOS}$ | 0.21 | 0.46 |

Cuadro 10.2: *MSE* y *RMSE* para calificaciones de usuarios y el propio estimador $\widehat{MOS}$.

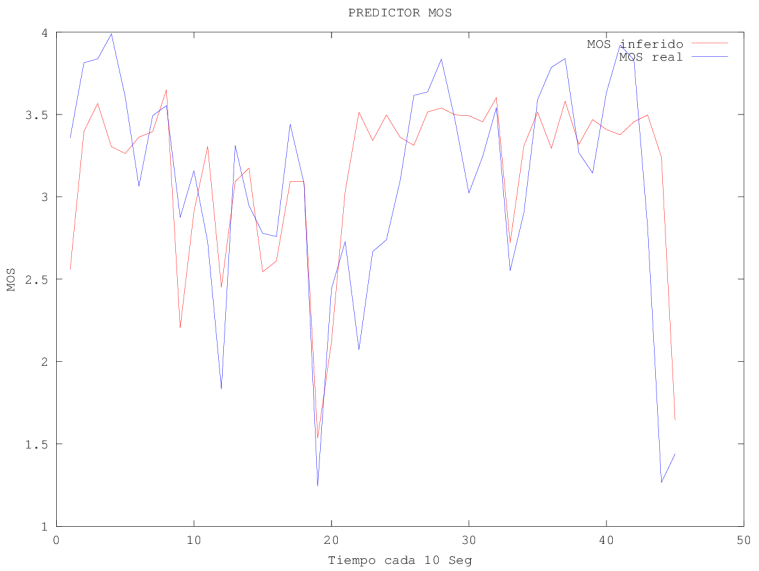


Figura 10.8: Valores obtenidos para $\widehat{MOS}$ y el valor real.

Se utiliza $\widehat{MOS}$ como una medida comparativa de eficiencia entre heurísticas. Una evaluación de calidad percibida, obtenida a partir de la caracterización que se hace dentro del reproductor de video sobre la disponibilidad de segmentos, comparte el mismo nivel de abstracción que la heurística. Este punto de contacto, tiene un interés especial porque permite medir el impacto en unidades de calidad percibida y comparar que grado de mejora hay entre una y otra.

Se compara el desempeño del sistema a partir de los valores de calidad percibida que se alcanzan en promedio, durante la operación del reproductor de video en el ensayo completo. Se requiere la utilización de los mismos escenarios, condiciones y contenido para cada prueba de heurística. Cada heurística procesa los escenarios impuestos como condiciones de acceso a los datos de origen, en forma típica, dando un conjunto de resultados característicos de su funcionamiento.

Se utiliza en esa parte la heurística original *JWPlayer original*, una modificación de la misma *JWPlayer positivo* y seis casos con representación fija *Fijo*, donde solo se consume la representación indicada según el Cuadro 9.5. En la Sección 6.3 se detalla el funcionamiento de *JWPlayer original*, mientras que en *JWPlayer positivo* es una modificación de la primera donde se quita la restricción vista en la Subsección 6.3.1, y se permite el incremento mayor a uno en el tipo de representación, quedando una heurística con una aproximación ”instantánea”. En el Cuadro 10.3 se resume el funcionamiento para cada una, donde se refleja su operación en general.

| Heurística | $\widehat{MOS}$ | Repres prom | Cant Det. | Det. seg |
|---|---|---|---|---|
| JWPlayer original | 3.15 | 6 | 27 | 5.8 |
| JWPlayer positivo | 3.35 | 7 | 39 | 4.5 |
| Fijo 8 | 2.6 | 8 | 141 | 6.2 |
| Fijo 7 | 2.96 | 7 | 115 | 3.6 |
| Fijo 6 | 3.31 | 6 | 7 | 4.6 |
| Fijo 5 | 3.12 | 5 | 1 | 14 |
| Fijo 4 | 3.11 | 4 | 0 | 0 |
| Fijo 3 | 2.98 | 3 | 0 | 0 |

Cuadro 10.3: Resultados de ensayos para cada heurística.

El mecanismo propuesto para obtener un estimador $\widehat{MOS}$ puede ser usado como herramienta de diagnóstico en segundos ensayos, con la repetición del mismo experimento en heurísticas diferentes, manteniendo sin cambio los escenarios y el contenido de prueba.

El propósito de estas pruebas es ver que mejora se logra en la experiencia del usuario, en el cambio de ajustes a realizarse en las heurísticas internas del reproductor de video. Cada heurística procesa los escenarios en forma típica. En el Cuadro 10.3, se resumen los resultados obtenidos.

A pesar del bajo número de participantes en el ensayo, se obtiene un estimador MOS razonable, de acuerdo al RMSE con respecto a MOS real, siendo comparable y menor en valor que la mejor evaluación de usuario. Los resultados obtenidos muestran un RMSE de 0.46 al MOS real del estimador, mientras que el mejor usuario evaluador del grupo obtiene 0.55.

# Capítulo 11

# Conclusión

## 11.1 Sobre la evolución del Streaming

La distribución de contenido de video en forma de *Stremaming* en Internet sobre protocolo *HTTP* es una tendencia que continúa en crecimiento. Es el resultado de la evolución de diferentes tecnologías y arquitecturas para la distribución y reproducción de *Video Streaming*. Los protocolos básicos, fueron incorporando con el tiempo nuevas características que permitieron funcionalidades ligeramente diferentes, dando nuevas posibilidades dentro de todo el sistema, lograron un mejor funcionamiento e hicieron más atractivo el uso para los consumidores, frente a los medios de *Broadcasting* tradicionales. Un subgrupo de estas tecnologías son los sistemas de *Video Streaming* adaptativos como se introduce en la Sección 3.2.3.

Estrictamente en las arquitecturas de tipo *HAS*, los servidores no utilizan recursos para administrar las descargas de sus consumidores, principalmente ofrecen y cumplen con requerimientos de descargas. Esto junto a las posibilidades de *Cache* disponible en Internet, permite más escalabilidad para este tipo de arquitecturas.

Por otro lado los reproductores de video logran, más grados de libertad para ajustar su consumo a una representación que mejor se adapte a sus restricciones ambientales. Las arquitecturas *HAS* utilizan en extremo la flexibilidad que da la segmentación del contenido, cuando se alinean los inicios de cada tramo en un *Key Frame*. En HAS, se ofrece a los consumidores diferentes opciones de codificación, que lo habilitan a elegir entre las disponibles y permiten con esta capacidad de adaptación atenuar el peor problema del *Video Streaming*, la continuidad del desplegado.

## 11.2 Sobre el funcionamiento de HAS

Los segmentos que contienen menos información son más fáciles de obtener, pero representan con más dificultad a la señal original y dan una experiencia baja en el desplegado. Este compromiso de mantener la continuidad frente a la exactitud de la señal original impacta en la calidad

percibida. Las heurísticas en HAS permiten de manera controlada degradar el desplegado a tasas conocidas de error, respecto a la señal original, en favor de mantener la continuidad, en resumen de la Sección 4.6.

Desplegar representaciones menos fieles a la señal original, da la posibilidad de atenuar el problema crítico de la detención. Mantener la continuidad y desplegar la mejor representación de la señal original disponible, son factores contrapuestos en este sistema. La selección del próximo segmento a descargar, como mecanismo de adaptación ambiental del reproductor, debe manejar el compromiso de la continuidad frente a la exactitud. Ambos factores impactan en la calidad percibida y la heurística los debe tener en cuenta para aplicar los criterios de selección.

## 11.3 Sobre la selección del próximo segmento

El objetivo principal de la heurística es superar el inconveniente de la detención, identificado siempre como el peor problema. En general las heurísticas utilizan el criterio de descargar la mejor representación que se pueda, restringido a que no ocurra vaciamiento del *Buffer* de video, como se muestra en resumen en la Sección 6.7.

Las heurísticas revisadas hacen mediciones de la capacidad de la red para obtener los segmentos, también predicen el largo del *Buffer* de video a futuro y estiman cual es la representación más conveniente para descargar. Este tipo de heurísticas no tienen en cuenta el impacto en la calidad percibida, sino que atenúan el problema de la detención. Toman como criterio que una mejor representación es equivalente a una mayor calidad percibida. En este trabajo se revisó el criterio anterior, ante la presencia de conmutaciones entre las representaciones, que también tiene impacto en la experiencia del usuario. Se analizó como evoluciona la calidad percibida en función los factores anteriores, con un enfoque general, para uno de los reproductores más utilizados.

## 11.4 Sobre el indicador de calidad percibida

El indicador equivalente a *MOS* se desarrolló usando un método de inferencia con aprendizaje, que relaciona la calidad percibida por los usuarios, con el acceso a los segmentos y sus calidades de representación. Estimar de esta forma la calidad percibida con un enfoque más directo que un ensayo *MOS*, permite obtener más económicamente un estimador comparable, $\widehat{MOS}$, con una precisión razonable, en resumen en la Sección 8.3.

El entrenamiento del método de inferencia se hace a partir de la evaluación MOS del ensayo y de variables locales registradas en simultáneo. Estas variables, son relativas a la disponibilidad y características de los segmentos que se procesan en las etapas de desplegado del reproductor, el punto más cercano al usuario dentro del sistema. Esto le da al mecanismo de estimación independencia de las capas subyacentes, que proveen los segmentos de contenido. Al mismo tiempo una evaluación con $\widehat{MOS}$, no involucra detalles acerca de constantes o valores típicos relativos a anchos de banda o cantidad de información a descargar. Es suficiente para

obtener $\widehat{MOS}$ las mediciones de disponibilidad para los segmentos, tipo de representación y largo de *Video Buffer* en tiempo de contenido. Este tipo de evaluación $\widehat{MOS}$, es ventajosa por la simplicidad de cálculo, por ser independiente de las tecnologías subyacentes y poder realizarse en tiempo real dentro del reproductor. Como principal desventaja, el cuantificador debe ser recalibrado si el contexto HAS cambia.

En el ensayo se verifica que el mecanismo propuesto de inferencia es suficiente para obtener un estimador $\widehat{MOS}$ y que es efectivo como cuantificador de la calidad percibida. Como criterio de validación se utiliza MSE con respecto a valores MOS reales, como se observa en el Cuadro 10.2 se obtiene el valor de 0.21 y su correspondiente RMSE. El RMSE puede ser usado como un estimador de la desviación estándar del $\widehat{MOS}$, sus unidades están en índice MOS y permiten hacer una comparativa con la evaluación de los usuarios. El valor obtenido de 0.46 es mejor aún, que el conseguido por el participante con menos variación al valor real.

Estos resultados impulsan la idea de que un ensayo con un mayor número de individuos, permite a partir de información disponible en el reproductor, estimar la calidad percibida por los usuarios. Queda la posibilidad de extender en el futuro, con una búsqueda de patrones similar a la que se presentó en este ensayo inicial, un mecanismo mejorado de patrones, para estimar MOS con más precisión.

## 11.5 Sobre la comparativa de heurísticas

Se aplicaron los mismos perfiles de perturbaciones externas a todo el sistema, para entender como se afecta a la calidad percibida el funcionamiento de las diferentes heurísticas para el mismo tipo de reproductor de video. Se utiliza el promedio del indicador propuesto ($\widehat{MOS}$) y ejecuta cada ensayo de heurística en iguales condiciones. En el Cuadro 10.3 se detalla, que heurística se utilizó y los valores promedio que se alcanzaron.

Como producto del funcionamiento de las heurísticas, se observan diferentes resultados en la calidad percibida, por lo que se concluye que algunas de estas son más eficientes para maximizar el objetivo que otras. Por ejemplo ligeras modificaciones en los algoritmos, como habilitar cambios en más de un nivel para el ascenso en el tipo de representación, determina una mejora respecto a la heurística original en la calidad percibida.

A diferencia de los otras arquitecturas de *Streaming* se verifica una influencia combinada en la calidad percibida, del tipo de representación desplegado y la continuidad. Una representación más fiel de contenido original, índices mayores para el Cuadro 9.5, aportan positivamente a la experiencia de usuario y por otro lado aumenta la posibilidad de que se generen detenciones, que tiende a ser una contribución negativa.

Como se puede observar para las pruebas de representación fija, en la Figura 11.1, hay un máximo característico que puede explicarse de este modo. El *Bit-Rate* de la representaciones

fija 6 y el *Bandwidth* efectivo promedio para el ensayo son comparables en magnitud. Concretamente el primero (1.2 mbps) es ligeramente menor que el segundo (1.83 mbps), lo que asegura un desplegado con menos interrupciones (y también de menos duración cada una), que la representación identificada como (7) en el Cuadro 9.5.

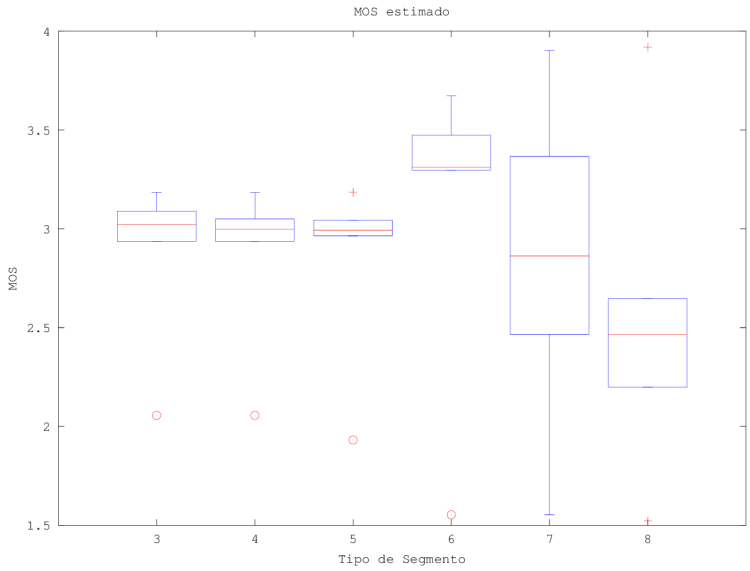


Figura 11.1: Calidad percibida estimada para representaciones fijas.

Se observa, que la operación de la heurística original (*JWPlayer original*), selecciona distintas representaciones, y se tiene un promedio de *6*, lográndose una calidad percibida de $\widehat{MOS} = 3.15$. Mientras que un reproductor sin dinámica, tomando en forma fija su próximo segmento a descargar, (*Fija 6*), tiene un evaluación de $\widehat{MOS} = 3.31$.

Los valores medios muestran una tendencia inicial para la calidad percibida frente a la influencia factores contrapuestos, como el tipo de representación y a las interrupciones. Además es consiste con el objetivo de las heurísticas en general, donde se busca descargar una representación más fiel restricto a no tener detenciones, en lugar de maximizar la calidad percibida por el usuario.

# Bibliografía


[1] Saamer Akhshabi, A. C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In *Proceedings of the second annual ACM conference on Multimedia systems*, pages 157–168. ACM, 2011.

[2] GEAPA Batista and Diego Furtado Silva. How k-nearest neighbor parameters affect its performance. In *X Argentine Symposium on Artificial Intelligence*, pages 95–106, 2009.

[3] Ali Begen, Tankut Akgul, and Mark Baugher. Watching video over the web: Part 1: Streaming protocols. volume 15, pages 54–63, Piscataway, NJ, USA, March 2011. IEEE Educational Activities Department.

[4] Ali C. Begen. IPTV, Internet Video and Adaptive Streaming Technologies. pages 1–100. Cisco Systems, 170 West Tasman Dr. San Jose, CA 95134 USA, 2012.

[5] J M Bonnin, G Rubino, and M Varela. Controlling Multimedia QoS in the Future Home Network Using the PSQA Metric. volume 2, 2006.

[6] Cristina García Cambronero and Irene Gómez Moreno. Algoritmos de aprendizaje: knn & kmeans. *Intelgencia en Redes de Comunicación, Universidad Carlos III de Madrid*, 2006.

[7] CONVIVA. Viewer Experience Report. pages 1–14. Silicon Valley, 2 Waters Park,Dr. Ste. 150, San Mateo, CA 94403, 2015.

[8] Microsoft Corporation. Advanced Systems Format (ASF) Specification. pages 1–124. Microsoft Corporation, Redmond, WA, USA, December 2004.

[9] Charles D. Cranor, Matthew Green, Chuck Kalmanek, David Shur, Sandeep Sibal, Jacobus E. Van der Merwe, and Cormac J. Sreenan. Enhanced streaming services in a content distribution network. *Internet Computing, IEEE*, 5(4):66–75, 2001.

[10] Amy Csizmar Dalal, Andy K. Bouchard, Sara Cantor, Yiran Guo, and Anya Johnson. Assessing QoE of on-demand TCP video streams in real time. In *Communications (ICC), 2012 IEEE International Conference on*, pages 1165–1170. IEEE, 2012.

[11] Luca De Cicco, Vito Caldaralo, Vittorio Palmisano, and Saverio Mascolo. ELASTIC: a Client-side Controller for Dynamic Adaptive Streaming over HTTP (DASH). In *Packet Video Workshop (PV), 2013 20th International*, pages 1–8. IEEE, 2013.

[12] Scripting Dictionary. ADOBE® DIRECTOR® 11.5.

[13] Wembo Zhang, Serge Smirnov, Kishore Kotteri, Gurpratap Virdi, Edgard Musayev, Florin Folta. Media Streaming Using an Index File. pages 1–29. Microsoft Corporation, Redmond, WA, USA, August 2008.

[14] Tobias Hoßfeld, Michael Seufert, Matthias Hirth, Thomas Zinner, Phuoc Tran-Gia, and Raimund Schatz. Quantification of YouTube QoE via Crowdsourcing. pages 494–499. IEEE, December 2011.

[15] V. Jacobson, R. Frederick, S. Casner, and H. Schulzrinne. RTP: A Transport Protocol for Real-Time Applications. pages 1–104. Network Working Group, Columbia University, 2003.

[16] Ukheon Jeong and Kwangsue Chung. Video Quality Adaptation to Improve The Quality of Experience in DASH Environments. *IJCSNS*, 14(8):22, 2014.

[17] Junchen Jiang, Vyas Sekar, and Hui Zhang. Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*, pages 97–108. ACM, 2012.

[18] Ingo Kofler, Robert Kuschnig, and Hermann Hellwagner. 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*, pages 549–553. IEEE, 2012.

[19] Xiaohua Lei, Xiuhua Jiang, and Caihong Wang. Design and implementation of streaming media processing software based on RTMP. In *Image and Signal Processing (CISP), 2012 5th International Congress on*, pages 192–196. IEEE, 2012.

[20] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3):67–82, 1997.

[21] B. Molina, C. E. Palau, M. Esteve, and J. Lloret. On content delivery network protocols and applications. *WSEAS Transactions on Computers*, 3(6):1981–1984, 2004.

[22] Sankar K Pal and Pabitra Mitra. Case generation using rough sets with fuzzy representation. *Knowledge and Data Engineering, IEEE Transactions on*, 16(3):293–300, 2004.

[23] L. E. Peterson. K-Nearest Neighbor. *Scholarpedia*, 4(2):1883, 2009. revision 136646.

[24] Ed. W. May R. Pantos. HTTP Live Streaming Overview. pages 1–42. Apple Inc., 2013.

[25] Benjamin Rainer, Stefan Lederer, Christopher Muller, and Christian Timmerer. A seamless Web integration of adaptive HTTP streaming. In *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European*, pages 1519–1523. IEEE, 2012.

[26] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data*. Department of Computer Science and Engineering, University of California, Riverside Riverside, CA 92521, 2004.

[27] David C. Robinson, Yves Jutras, and Viorel Craciun. Subjective Video Quality Assessment of HTTP Adaptive Streaming Technologies. volume 16, pages 5–23. Bell Labs Technical Journal, March 2012.

[28] Demóstenes Z. Rodríguez, Zhou Wang, Renata L. Rosa, and Graça Bressan. The impact of video-quality-level switching on user quality of experience in dynamic adaptive streaming over HTTP. *EURASIP Journal on Wireless Communications and Networking*, 2014(1):216, 2014.

[29] Sumit Roy, Michele Covell, John Ankcorn, Susie Wee, and Takeshi Yoshimura. A system architecture for managing mobile streaming media services. In *Distributed Computing Systems Workshops, 2003. Proceedings. 23rd International Conference on*, pages 408–413. IEEE, 2003.

[30] ITU-T Telecomunication Standardization Sector. Subjective video quality assessment methods for multimedia applications. pages 1–37. INTERNATIONAL TELECOMMUNICATION UNION, 1999.

[31] ITU Telecommunication Standardization Sector. Measurement of the quality of service. pages 1–10. TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 1996.

[32] Margaret Pinson Stephen Wolf. Reference Algorithm for Computing Peak Signal to Noise Ratio (PSNR) of a Video Sequence with a Constant Delay. pages 1–18. NTIA/ITS (USA), February 2-6, 2009, 2009.

[33] Adobe Systems. Adobe Media Manifest Specification. In *Flash Media Manifest (F4M) Format Specification*, pages 1–45. adobe.com, 2013.

[34] Cisco Systems. Cisco Visual Networking Index: Forecast and Methodology, 2014–2019. pages 1–10. Cisco Systems, 170 West Tasman Dr. San Jose, CA 95134 USA, 2014.

[35] Stockhammer Thomas. Dynamic Adaptive Streaming over HTTP – Design Principles and Standards. volume 2014, pages 1–5. MPEG Dynamic Adaptive Streaming over HTTP, 2014.

[36] Christian Timmerer and C. Müller. HTTP Streaming of MPEG Media. pages 1–4. Information Technology (ITEC), Klagenfurt University, Klagenfurt, Austria, 2010.

[37] International Telecommunication Union. RECOMMENDATION ITU-R BT.500-13 - Methodology for the subjective assessment of the quality of television pictures - R-REC-BT.500-13-201201-I. pages 16–23. ITU-R, 2012.

[38] Ezequiel Uriel. Regresión lineal múltiple: estimación y propiedades. *Universidad de Valencia Versión*, pages 09–2013, 2013.

[39] Qingyang Wang, Deepal Jayasinghe, and Pengcheng Xiong. ENet-Dynamic QoS Controller for Video Streaming Application. pages 1–4. College of Computing, Georgia Institute of Technology, Atlanta, USA, 2012.

[40] Xindong Wu, Vipin Kumar, J. Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J. McLachlan, Angus Ng, Bing Liu, Philip S. Yu, Zhi-Hua Zhou, Michael Steinbach, David J. Hand, and Dan Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, January 2008.

[41] Hongyun Yang, Xuhui Chen, Zongkai Yang, Xiaoliang Zhu, and Yi Chen. Opportunities and Challenges of HTTP Adaptive Streaming. *International Journal of Future Generation Communication & Networking*, 7(6), 2014.

[42] Alex Zambelli. IIS Smooth Streaming Technical Overview. pages 1–29. Microsoft Corporation, Redmond, WA, USA, 2009.

[43] Yingnan Zhu, Wei Liu, Lina Dong, Wenjun Zeng, and Heather Yu. High performance adaptive video services based on bitstream switching for IPTV systems. In *Consumer Communications and Networking Conference, 2009. CCNC 2009. 6th IEEE*, pages 1–5. IEEE, 2009.

# Glosario

Diccionario de abreviaturas:

Bandwidth: Ancho de Danda
Buffer: Almacenamiento intermedio
Cache: Almacenamiento intermedio
CDN: Red para difusión de contenido
CODEC: Codificador/Decodificador de video
DTW: Envoltura de tiempo dinámico
FPS: Cuadros por segundo
Frame: Cuadro de imagen
GOP: Grupo de imágenes
HAS: Sistema adaptativo de HTTP Streaming
HDTV: Formato de alta resolución para TV
HTTP: Protocolo de transferencia de hipertexto
HTTP Streaming: Internet Video
IEC: International Electro Technical Commission
IP: Protocolo de Internet
IPTV: Televisión por Internet
ISO: International Organization for Standard
ITU-R: International Telecommunication Union, Radio Communication Sector
ITU-T: International Telecommunication Union, Telecommunication Standardization Sector
LIVE: Contenido en vivo
MOS: Calificación media de encuesta
MPEG: Moving Pictures Expert Group
Multibitrate: Capacidad para cambio de tasa de bits
Pixel: Unidad de imagen
PSNR: Relación máxima señal ruido
QoE: Calidad de la experiencia
QoS: Calidad de servicio
Representación: Contenido codificado en datos, de la señal original
RTMP: Protocolo de mensajes en tiempo real
RTP: Protocolo de transporte en tiempo real
RTSP: Protocolo de transferencia en tiempo real
SDTV: Formato de resolución Standard para TV

Streaming: Transferencia continua de información
VOD: Contenido a Demanda
VQEG: Video Quality Expert Group

# Índice de figuras