UCLV
Universidad Central
"Marta Abreu" de Las Villas

FIE
Facultad de
Ingeniería Eléctrica

Departamento de electrónica y telecomunicaciones

TRABAJO DE DIPLOMA

Título: Servicio *streaming* de video con codificación H.265

Autor: Rudenz Reinel Cruz Ramírez

Tutor: Rafael Alejandro Olivera Solis

Santa Clara, Junio, 2019

UCLV
Universidad Central
"Marta Abreu" de Las Villas

FIE
Facultad de
Ingeniería Eléctrica

Telecommunications and Electronic Academic Departament

TRABAJO DE DIPLOMA

Title: Video streaming service with H.265 encoding

Author: Rudenz Reinel Cruz Ramírez

Thesis Director: Rafael Alejandro Olivera Solis

Santa Clara, June, 2019

Hago constar que el presente trabajo de diploma fue realizado en la Universidad Central "Marta Abreu" de Las Villas como parte de la culminación de estudios de la especialidad de Ingeniería en Automática, autorizando a que el mismo sea utilizado por la Institución, para los fines que estime conveniente, tanto de forma parcial como total y que además no podrá ser presentado en eventos, ni publicados sin autorización de la Universidad.

Firma del Autor

Los abajo firmantes certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro y el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

Firma del Tutor

Firma del Jefe de Departamento
donde se defiende el trabajo

Firma del Responsable de
Información Científico-Técnica

PENSAMIENTO

La tecnología no es nada. Lo importante es que tengas fe en la gente, que sean básicamente buenas e inteligentes, y si les das herramientas, harán cosas maravillosas con ellas.

Steve Jobs.

DEDICATORIA

A mi mamá,

Por ser la mejor de todas, por ser la que siempre ha estado junto a mí toda mi vida, por todos sus consejos y su apoyo incondicional. Gracias por todo su amor y comprensión. Gracias por darme la vida y por hacer que esta sea la mejor.

A mi papá,

Por siempre apoyarme por estar en todos los momentos buenos y malos, por todo su amor y por todas sus enseñanzas, por ser su hijo.

A mi hermana,

Por ser mi mayor tesoro, por siempre verle el lado bueno a las cosas, por siempre estar allí para mí y por enseñarme tanto de la vida a pesar de ser más pequeña que yo.

A mis abuelas,

Por todo su amor y cariño, por siempre pensar en mí, por darme la mejor de todas las educaciones para la vida y por darme una maravillosa niñez, por malcriarme tanto, gracias por siempre estar.

A toda mi familia, por siempre apoyarme y estar ahí.

A todos mis amigos por las buenas y malas noches que hemos pasado, por todo su apoyo incondicional.

AGRADECIMIENTOS

A mi mamá, por su constante sacrificio para que yo cumpla todos mis sueños, por estar conmigo en cada paso que doy, por su educación, tanto académica como para la vida, por todos sus consejos.

A mi papá, por ser un ejemplo de padre en todo momento del cual he aprendido muchísimo, por los valores y principios que me ha inculcado.

A mi hermana, por hacerme una mejor persona y un mejor hermano

A mi familia: a mis abuelas: Silvia, Esperanza y Nena, a mis tíos: Ernesto, Milagros, Marisela, Ernestina a mis primos

A mis amigos, sobre todo, los más íntimos, tanto en la universidad como fuera, por la amistad que me han brindado en especial a Adianis, Cabeza, Rafael, Carlos el Loco, Elier, José Carlos, Alian, Adalberto por los momentos que hemos pasado a lo largo de estos años que nunca olvidaré.

A mi tutor Rafael, por su ayuda incondicional, no solo en la elaboración de la tesis sino también a lo largo de la carrera, por no solo haber compartido sus conocimientos, sino también su amistad.

A todos los que hicieron posible esta meta, gracias por todo.

TAREA TÉCNICA

1. La revisión de la bibliografía técnico – especializada relacionada con las tecnologías de *streaming* y de herramientas de compresión de la información.
2. Caracterización de las arquitecturas del *streaming* de video.
3. Caracterización de técnicas de compresión eficiente.
4. Implementación de diferentes variantes para el servicio *streaming*
5. Implementación de diferentes variantes de acceso al servicio *streaming*.

Firma del Autor

Firma del Tutor

RESUMEN

En este trabajo se aborda el desarrollo de un servidor *streaming* de video en tiempo real con codificación HEVC con el uso de herramientas de *open sources*. Durante la elaboración de este trabajo se exponen distintas arquitecturas de *streaming*, los servicios que este ofrece y se profundiza sobre las diferentes tecnologías en las que se pueden implementar el *streaming* de video. También se describen cómo utilizar ***FFmpeg*** y cómo en él se pueden ajustar parámetros de codificación y decodificación de video para optimizar el trabajo con la compresión H.265/HEVC con el fin de lograr altas velocidades de codificación y eficiencia de compresión ahorrando grandes cantidades de ciclos de CPU. Posteriormente, se describe cómo realizar un *streaming en tiempo real* y cómo facilitar el servicio a los usuarios al acceder a la transmisión en tiempo real auxiliándose de un servidor *apache*. Este tema tiene una gran importancia para el desarrollo tecnológico en La Universidad Central Marta Abreu de las Villas. Además, de ser una guía para posteriores trabajos referentes al tema en los que se puede validar dicho servicio.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Cuando se habla de telecomunicaciones, se refiere al transporte de información a grandes distancias a través de algún medio o canal de distribución utilizando señales de cualquier tipo. Desde el surgimiento del teléfono las telecomunicaciones han evolucionao tanto que en la actualidad no solo se puede escuchar a otra persona que esté en un punto distante sino que se puede escuchar y ver sin necesidad de estar cerca; buscar cualquier tipo de informacion en Internet, enviar correos electrónicos y hasta realizar *streaming* de audio y video desde y hasta cualquier parte del mundo.

En la actualidad el usuario busca servicios de video con la mejor calidad para tener la mejor experiencia en la visualización de imagen actualmente existen codificadores de video, que garantizan mayor calidad de imagen a menores razones de bits, entre ellos están VP8, VP9, H.264/*AVC* (*Advanced Video Códec*) y H.265/HEVC.

HEVC (High Efficiency Video Coding) es un estándar de codificación de video que fue desarrollado conjuntamente por las organizaciones ITU-T VCEG (*Video Coding Experts Group*) e ISO/IEC *MPEG* (*Moving Picture Experts Group*). HEVC es también conocido como ISO/IEC 23008-2 *MPEG*-H Part 2 o ITU-T H.265. Su objetivo principal es mejorar la compresión de video, en relación a los anteriores estándares (H.264/*MPEG-4 AVC*). HEVC puede soportar 8K (Ultra High Definition video), con una resolución de hasta 8192x4320 píxeles. HEVC cuenta con *x265* que es una aplicación y biblioteca de software libre y código abierto para la codificación de video que utiliza esta norma. Desde el surgimiento del video siempre se ha buscado la forma de reducir el *bitrate* del video transmitido, y a su vez, mantener su calidad lo mayor posible, trayendo consigo la creación de herramientas que permitan un proceso de codificación eficiente y con mayor calidad.

La primera patente que utilizó la palabra *streaming* fue en los años 20, cuando la empresa Muzak desarrolló una plataforma de música continua para negocios, este dato es interesante ya que ni siquiera existían computadoras en ese entonces.

Parte importante en la historia del *streaming* fue la evolución del Internet, así como de las conexiones de banda ancha, ya que los primeros accesos a la red eran hechos a través de líneas de teléfono convencionales, lo máximo que se podía conseguir de velocidad eran 27 kbps y era prácticamente imposible hacer transmisiones en tiempo real, pero con el crecimiento y evolución de nuevas infraestructuras como DSL o fibra óptica, y la comercialización del Internet con costos cada vez más accesibles para las personas, la velocidad del Internet dejó de ser un problema y es ahí cuando el *streaming* vio su oportunidad real de ser un estándar como lo es hoy en día.

Las estaciones de radio por Internet fueron el primer boom del *streaming*, ya que no se necesitaba tanta velocidad para sintonizar el audio de manera fluida, era lógico pensar que su metamorfosis iba encaminada al video, cuando las primeras transmisiones en tiempo real salieron a flote, sólo eran vistas por 2 ó 3 personas, ya que no había la capacidad de hacerlo escalable, y eran eventos de máximo 4 horas.

En 1997 la banda Severe Tire Damage hizo historia al transmitir su concierto en vivo a todo el mundo, esto fue en el Xerox PARC, más tarde en 1995 la empresa Real Networks transmitió por primera vez en *streaming* un juego de béisbol de los playoffs, ese mismo año en Seattle se realizó la transmisión de un concierto de la sinfónica en el Paramount Theater.

La comercialización del *streaming* tuvo que esperar varios años ya que va de la mano con los avances tecnológicos y el crecimiento exponencial del uso del Internet y de las capacidades en hardware de las computadoras, posteriormente computadoras más veloces y un Internet más rápido resolvieron la ecuación.

El año 2005 fue muy importante para la proliferación de los videos gracias al lanzamiento del portal de Youtube, además de la creación de los reproductores de videos basados en flash.

Gracias al desarrollo de los servicios *streaming* existen plataformas como Netflix, Spotify, Youtube, CWS entre otras, donde cualquier usuario con conexión a Internet puede poner en su sitio web a reproducir un video en vivo en sólo unos segundos. Con el desarrollo de

HEVC se puede enviar videos en 1080p o 4K y pueden ser vistos con conexiones caseras de Internet sin temor a cortes.

Lo único certero con el *streaming* es que no se va a ir a ninguna parte, ya que incluso cambió la manera en cómo se consume los contenidos de video como la TV tradicional.

Acerca de este tema se pueden encontrar muchos trabajos:

En [1] se implementa un servicio web para enviar y reproducir contenido multimedia por medio de la tecnología *streaming*. Enfocada en el uso para comunidades de bajos recursos los cuales tengan acceso restringido o escaso a fuentes de información.

En [2] se desarrollar e integrar a una plataforma el monitoreo de vehículos por medio de video que permita a las industrias, empresas y pymes el control, vigilancia y seguimiento de los recursos físicos de aquellas instituciones, los procesos que se realicen en estos y el correcto uso por parte del personal encargado ya que actualmente no son muchos los productos en el mercado con la versatilidad del video en tiempo real, solamente se puede hacer seguimiento de ubicación, pero sin detallar en información de video en tiempo real.

En [3] se estudia de la implementación de un servidor de video *streaming* para su despliegue en un sistema de gestión de aprendizaje para la carrera de Ingeniería de Electrónica y Telecomunicaciones de la Facultad de la Energía de la Universidad Nacional de Loja, esto con el propósito de mejorar el proceso de aprendizaje de modo que facilite las actividades tanto a los docentes como a los estudiantes.

En la Universidad Marta Abreu de Las Villas no han sido poco los trabajos de investigación que se han hecho en los últimos años relacionados con el tema.

En [4] se realiza un análisis de la calidad perceptual de video utilizando el estándar de compresión H.265/HEVC, donde se demostró que H.265 logra una calidad de video similar a la de H.264 y solo necesita la mitad de la razón de bit.

En [5] se utiliza la herramienta ***FFmpeg*** (*Fast Forward Moving Picture Expert Group*) para la codificación de materiales audiovisuales usando H.265 con la cual se puede optimizar los servicios de almacenamiento y reproducción de video. Se realizó una síntesis de los formatos de compresión video más importantes desde sus inicios hasta la actualidad. Se analiza el *códec* H.265 y las métricas PSNR y SSIM para medir la calidad de la

codificación eficiente. Además, se codifican materiales audiovisuales utilizando herramientas de código abierto y utilizando el HPC (*High performance Computing*) de la Universidad Central Marta Abreu de Las Villas.

En [6] se realizó un estudio comparativo de los *códec* H.264 y H.265 basado en métricas objetivas de calidad de video, se analizó la teoría de los codificadores, las principales métricas de calidad de video, los parámetros para las codificaciones, así como las herramientas de *software* a utilizar para las pruebas.

Actualmente es muy frecuente que cualquier organismo cuente con una base de datos con su material audiovisual en formato digital para su distribución en Internet. La universidad Marta Abreu de Las Villas no cuenta con una plataforma en la que se pueda aprovechar las redes de comunicaciones para hacer un *streaming* de video a bajas razones de transmisión, debido a esto se plantea la siguiente interrogante científica: ¿Cómo contribuir al desarrollo tecnológico audiovisual de la Universidad Central "Marta Abreu" de las Villas utilizando *open sources*?

Objetivo General:

Implementar un servicio de *streaming* de video utilizando ***FFmpeg*** y el estándar de compresión H.265/HEVC.

Objetivos específicos:

1. Caracterizar las tecnologías asociadas al *streaming* de video.
2. Describir el servicio *streaming* y las herramientas necesarias para su realización.
3. Proponer variantes de configuración para el servidor *streaming* respetando las regulaciones existentes para un campo universitario.

El informe de la investigación se conformará de introducción, capitulario, conclusiones, recomendaciones, referencias bibliográficas y anexos. En el capítulo 1 se caracterizará el *streaming* de video y las tecnologías asociadas a él, en el capítulo 2 Se expondrá como sera el servicio de *streaming* en la UCLV y se describirán las herramientas necesarias para su realización, en el capítulo 3 se implementa el servicio de video *streaming* en un servidor de la universidad para ver su funcionamiento. A partir de un análisis crítico de los resultados, se muestran las conclusiones de la investigación desarrollada, así como también las

recomendaciones en función de futuras investigaciones sobre el tema. Las referencias bibliográficas se relacionan posteriormente, mostrando que el tema es de actualidad y cuenta con marcada novedad. Seguidamente se relacionan los anexos.

Capítulo 1. *Streaming* de video

1.1 Introducción

En este capítulo se caracterizará el *streaming* de video, sus servicios asociados, la estructura para su realización y se hará una comparación entre las principales tecnologías con las que se puede implementar. Además, se analizará el funcionamiento de algunas plataformas de *streaming*.

1.2 *Streaming* de video. Definiciones

El uso de *streaming* de video se ha tornado sumamente popular. En el mercado se encuentran una multitud de terminales (móviles, *Smart TV's, tablets*, laptops). La mejora en los provedores de servicio, tanto de servicios móviles como ISP (Internet *Services Provider*), facilitan que se acceda a videos en Internet desde cualquier lugar. Cada día el tráfico que genera el servicio *streaming* se incrementa exponencialmente llegando a ser, junto con el acceso a Internet, los principales referentes en cuanto a consumo de ancho de banda. La primera consideración a tener cuando se diseña una red de comunicaciones es el entorno, donde el usuario cada vez exige continuas mejoras en los servicios: más rapidez, menos errores (o problemas), mejores prestaciones y más calidad. Es donde nace la necesidad de desarrollar continuamente protocolos que permitan utilizar el ancho de banda del cliente de una manera eficiente, sin perjuicios de otras aplicaciones.

El *streaming* engloba un conjunto de productos y técnicas cuyo objetivo es la difusión de contenidos multimedia tales como audio y video. Su orientación está dirigida absolutamente para su utilización en Internet. Es necesario reducir el tamaño del video y comprimirlo para su almacenamiento en un servidor web [7].

En la navegación por Internet tradicional es necesario descargar previamente el archivo (página HTML (*Hyper Text Markup Language*), imagen JPG (*Joint Photographic Experts Group*), audio MP3(*MPEG-1 Audio Layer III*), video *MPEG* (*Moving Picture Experts Group*), desde el servidor remoto al cliente local para luego visualizarlo en la pantalla de este último [8].

Una solución parcial a este problema es el flujo continuo de audio/video (*stream*), llamado comúnmente *streaming*, donde continuamente se solicitan datos de video, o de audio, al servidor y no se espera a que lleguen todos para poder ver las imágenes u oír el sonido en el lado del cliente, sino que se reproduce el video y/o audio conforme llegan los datos que lo componen [7].

El *streaming* es una tecnología de transmisión a través de la red, en la que no existe descarga de la información en un disco local y que, por ende, la información que se envía a través de la red al cliente se reproduce en tiempo real al recibirla. En la figura 1.1 se aprecian los elementos del servicio *streaming*. La cuestión es que un stream debe ser transmitido de modo que cualquiera pueda conectar con él en cualquier momento, y no sólo al principio de la transmisión [9].

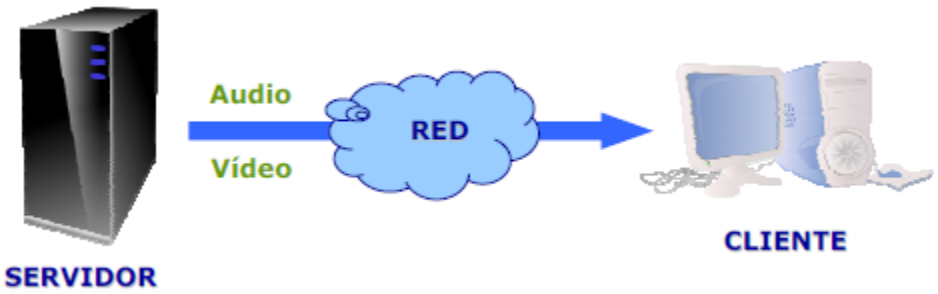


Figura 1.1 Elementos de un servicio de *streaming* [10].

En la realización del *streaming* se debe tener en cuenta los códec de video, puesto que son una parte importante dentro de este proceso, dado que ayudan a reducir el espacio de almacenamiento y aprovecharlo eficientemente.

Se puede considerar principalmente estos tres factores para el servicio de *streaming*:

- El formato adecuado
- El servidor de video-*streaming*

- El ancho de banda

El uso de un servidor es un aspecto importante que se debe mencionar, puesto que se necesita un servidor para almacenar el archivo multimedia y un software que permita la transmisión de los datos a través de la web. Así, la elección más adecuada de estos elementos es un aspecto muy significativo porque si el servidor no posee los suficientes recursos de cómputo para hacer frente a las demandas de los usuarios, estos recibirán una mala calidad de video y con ello imágenes degradadas, sonidos bruscos e incluso la pérdida de la conexión.

1.2.1 Funcionamiento del *streaming*

Conexión con el servidor. El reproductor cliente se conecta con el servidor remoto y éste comienza a enviarle información multimedia.

Buffer. El cliente empieza a recibir el fichero y construye un *buffer* o almacén donde comienza a guardarlo.

Inicio de la reproducción. Cuando el *buffer* se ha llenado con una pequeña fracción inicial del archivo original, el reproductor cliente comienza a mostrarlo mientras continúa en segundo plano con el resto de la descarga.

Caídas de la velocidad de conexión. Si la conexión experimenta ligeros descensos de velocidad durante la reproducción, el cliente podría seguir mostrando el contenido consumiendo la información almacenada en el *buffer*. Si se vacía el *buffer* se detendría la visualización hasta que se vuelva a llenar el mismo [11],[8].

1.2.2 Clasificación del *streaming*.

Descarga progresiva

Todos los servidores web son capaces de descarga progresiva. Esto es simplemente el método de un archivo de video que se entregarán a través de HTTP (*Hypertext Transfer Protocol*) a un navegador. Esto es similar a alguien que descarga un archivo desde su sitio web. De hecho, el video se entrega de la misma manera que una imagen, CSS, un JS, PDF o cualquier otro archivo de su sitio web.

La diferencia real es que los reproductores multimedia pueden empezar a mostrar el video mientras se está descargando. Por ejemplo, un archivo FLV entregado a través de la descarga progresiva de HTTP comenzará a reproducir en su reproductor Flash al recibir los primeros datos del flujo de video en el navegador.

Es bastante obvio cuando un video se entrega mediante descarga progresiva de HTTP. Normalmente, se verá la barra de estado pequeño crecer como las descargas de video. Y el usuario no será capaz de mover la tecla de desplazamiento más allá de la cantidad que ya se ha descargado. Esto hace que sea imposible saltar al final del video si solo se ha descargado la primera parte. Si se tiene un servidor web de ancho de banda lento o limitado o el usuario final está en una conexión a Internet lenta, entonces es posible que se detenga la reproducción.

El *buffering* se produce cuando la descarga no puede mantenerse por delante de video. El video se interrumpe durante la descarga. Si se detiene el video y le permite descargar una gran parte, a continuación, puede ver el video ininterrumpido. En ambos casos, se trata de una experiencia de usuario final deficiente, esto es cuando se debe considerar el uso de un CDN (Red de Distribución de Contenidos) [12].

Un CDN es una infraestructura informática en la que se entrelazan varios ordenadores distribuidos geográficamente en varios data centers, en la figura 1.2 se aprecia una red CDN. Estos almacenan parte de la información y el contenido de los sitios web y los servirán al usuario final.

Sus ventajas son que mejoran la disponibilidad del servidor, alivian la carga de tráfico de este, son una barrera más de seguridad contra ataques informáticos, y además mejoran el rendimiento y los tiempos de carga [13].

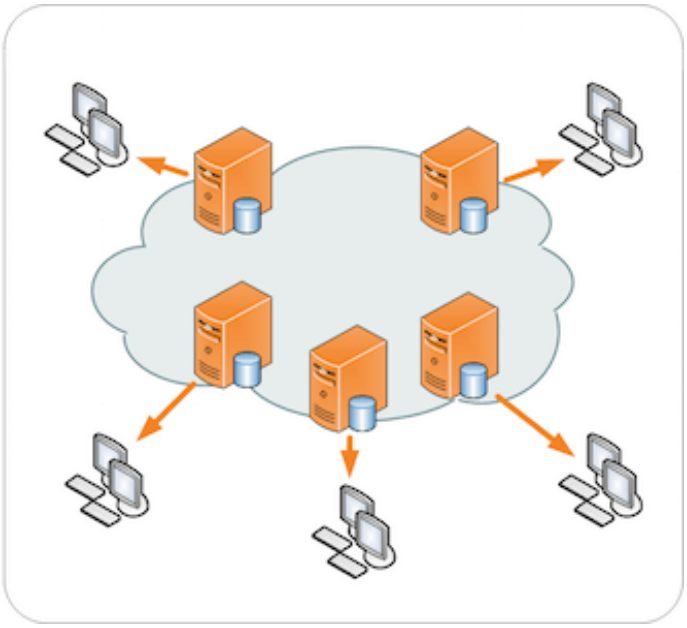


figura 1.2 Estructura CDN [13].

Cuando un video se entrega a través de HTTP, en realidad es descargado en los ordenadores de los usuarios finales. Esto es bueno y malo. Bueno porque si la persona quiere ver el video de nuevo, ya se almacenan en caché en el equipo, y malo porque hace extremadamente fácil para alguien de robar su contenido.

Si alguien mira sólo el primer minuto del video, pero no se detiene la descarga, el navegador descarga el archivo completo y el cliente va a pagar por la entrega del archivo, incluso si la persona no lo ve en absoluto. [12].

Transmisión por secuencias. Se produce en servidores multimedia que disponen de un *software* especial para gestionar correctamente el *streaming* de audio y video: *Windows Media Server*, *Flash Communication Server*. La utilización de un servidor multimedia ofrece múltiples ventajas frente al servidor web. Las más destacadas son:

- Mayor rapidez en la visualización de este tipo de contenidos.
- La comunicación entre servidor/cliente se puede realizar por protocolos alternativos al HTTP. Tiene el inconveniente del bloqueo impuesto por *firewalls*, pero tiene la ventaja de una mayor rapidez.
- Mejor gestión del procesador y ancho de banda de la máquina del servidor ante peticiones simultáneas de varios clientes del mismo archivo de audio o video.

- Control predefinido sobre la descarga que pueden realizar los clientes: autentificada, filtrada por Ip, sin almacenarla en la caché del cliente.
- Mayor garantía de una reproducción ininterrumpida gracias al establecimiento de una conexión de control inteligente entre servidor y cliente.
- Posibilidad de distribución de transmisiones de audio y video en directo [8] [14].

En la figura 1.3 se observan los procesos inherentes al *streaming* de video.

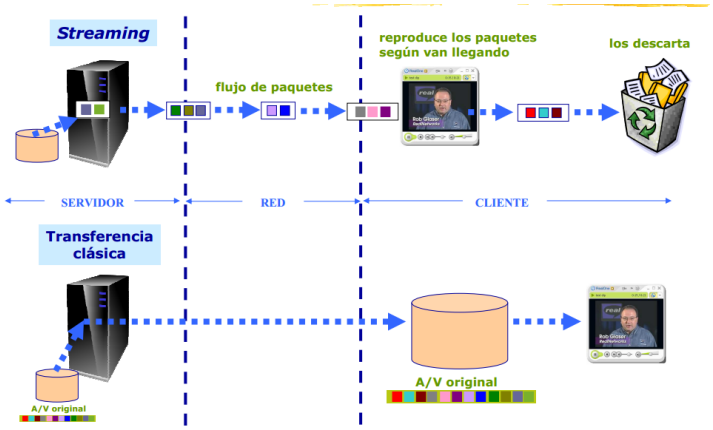


Figura 1.3 *Streaming* de video [10].

1.3 Tipos de Servicios

El proceso de *streaming* se puede dividir en dos categorías, en función de cómo se obtiene la información a difundir: *streaming* en directo (similar a un canal de televisión) o bajo demanda (similar a un reproductor de video).

1.3.1 *Streaming* en directo

El *streaming* en directo, como se aprecia en la figura 1.4, es aquel que transmite eventos que están sucediendo justo en el momento de la difusión. Por ejemplo, la transmisión de conciertos o de clases son eventos que típicamente se difunden usando este tipo de *streaming*. La transmisión de radio y televisión por Internet también tiene estas características, aunque en ocasiones parte de la información que se difunde no parte de un evento en directo (por ejemplo, un programa que ha sido grabado previamente, pero que se va a difundir en un momento determinado) [10].

En este tipo de transmisión se emplea el término difusión (*broadcast*) porque realmente se está transmitiendo "en vivo" a todos los clientes la misma información, que no es más que el evento que se está produciendo en ese momento. Así, independientemente de cuando se conecta un cliente al servidor, todos ven exactamente el mismo punto del *stream* en un instante determinado (excepto las lógicas variaciones de los retardos en la red que hacen que unos clientes reciban antes los datos que otros). Para poder efectuar este tipo de transmisión no es suficiente con disponer de un servidor de *streaming*, sino que también es necesario un equipo que realice el proceso de captura y compresión en tiempo real (que a veces se conoce como difusor o *broadcaster*). Este equipo puede estar instalado en la misma máquina que el servidor de *streaming* si el número potencial de clientes no es grande, pero para resultados profesionales, en un entorno con muchos clientes, es conveniente separar ambos programas en dos servidores distintos. Además, para dar un servicio realmente eficiente de este tipo de *streaming* es conveniente que la difusión se realice con técnicas de *multicast* [15], [16].

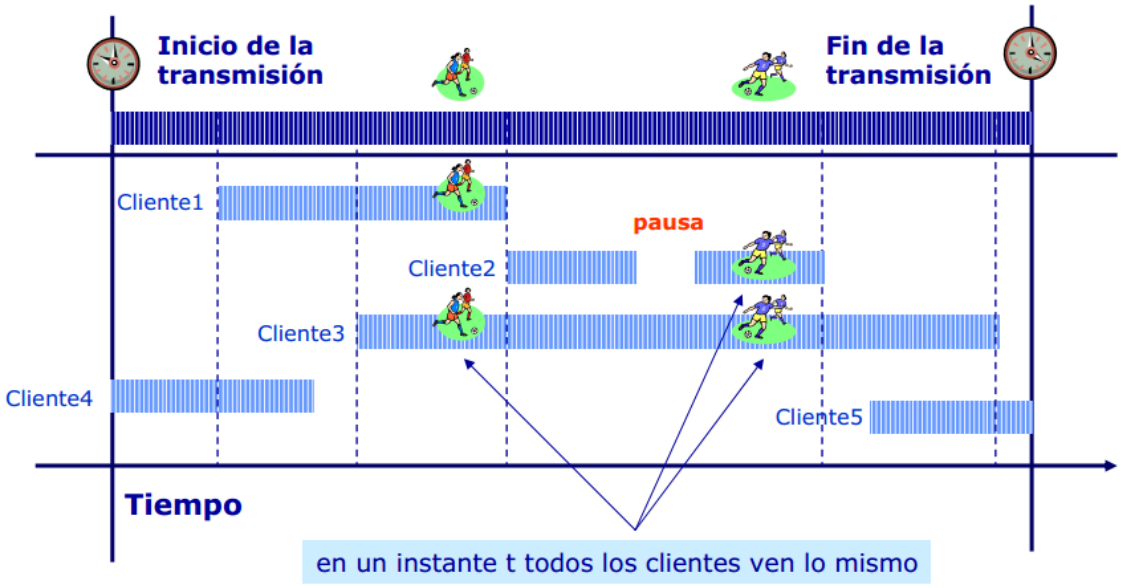


Figura 1.4 Ejemplo de *streaming* en directo [10].

Según el origen de las señales de audio/video el *streaming* en directo puede ser:

- Con información en vivo.
- Con información almacenada.

Según el tipo de transmisión el *streaming* en directo puede ser:

- ***Unicast*** : se envía un flujo de información a cada usuario.
- ***Multicast*** : se envía un flujo único de información [10],[15].

1.3.2 *Streaming* bajo demanda:

En un *stream* multimedia bajo demanda, como se aprecia en la figura 1.5, la transmisión del medio empieza desde el inicio del evento a ser reproducido para cada uno de los clientes. El medio a transmitir puede estar ya preparado desde el comienzo del proceso en un fichero comprimido. En este caso no representa una ventaja adicional el disponer de la posibilidad de realizar *multicast* en la red, ya que cada cliente recibe una parte distinta del *stream* y por lo tanto un paquete de datos diferente [17],[18].

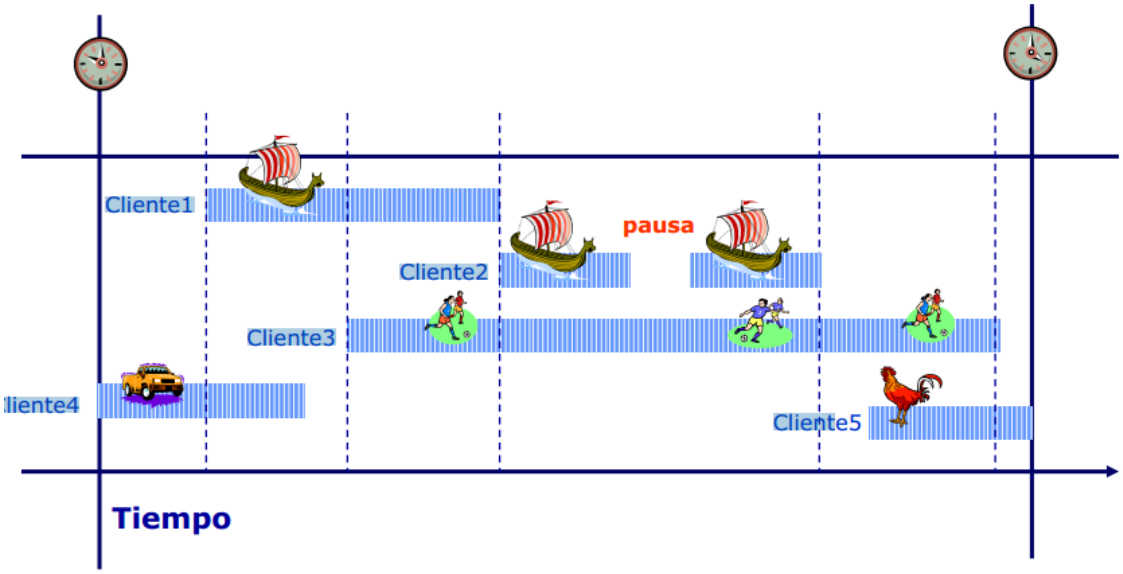


Figura 1.5 Ejemplo de *streaming* bajo demanda [15].

1.4 Arquitectura del *streaming* de video

Para realizar un *streaming* de video se pueden utilizar diferentes arquitecturas, cada una tiene sus ventajas y sus desventajas, y se utilizan según el objetivo que se persiga. Cada una de ellas cuenta con una serie de elementos que se aprecian en la figura 1.6.

1.4.1 Tipos de Arquitecturas

1 Arquitectura típica (con servidor y cliente)

2 Arquitectura sin servidor (*server-less*)

- No hay servidor de audio y video.
- Se sirve mediante un servidor web
- Da lugar a servicios de *pseudo-streaming* o *fast-start*

3. Arquitectura sin cliente (*client-less*)

- No hay programa cliente
- Se utiliza para visualizar un *applet* Java o un *plugin* (p.e. Flash)

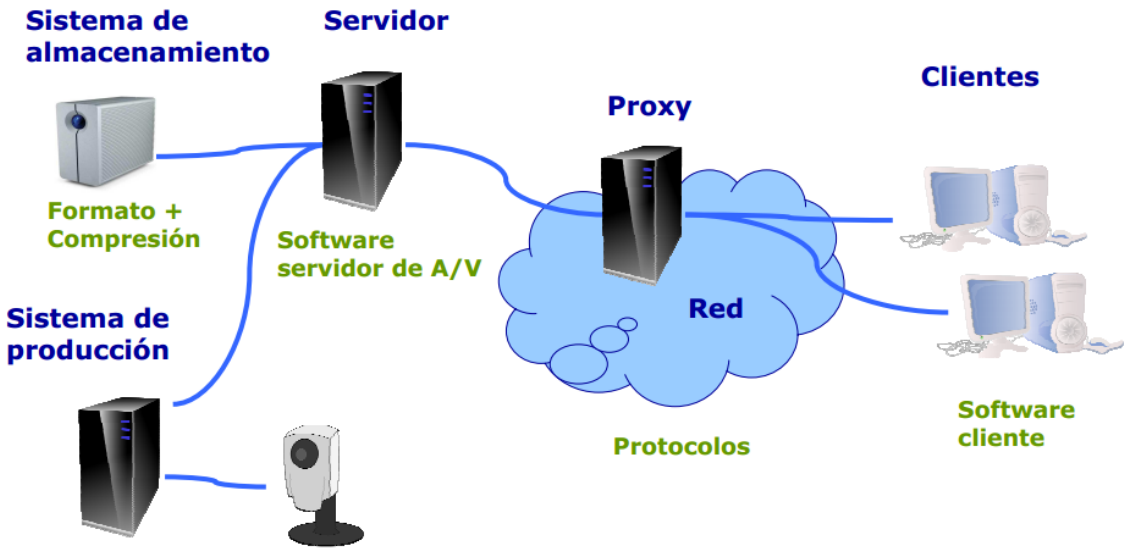


Figura 1.6 Arquitectura típica de un *streaming* de video [10].

1.4.2 *Streaming* típico o tradicional

Los elementos que forman un *streaming* tradicional son:

Sistema de producción

Genera los flujos de audio/video que se van a transmitir, existen dos tipos de producción: para almacenar y para emitir en directo, el *hardware* está formado por elementos de

adquisición (cámaras, micrófonos, capturadoras), y el *software* se divide en *software* de edición y *software* de producción para transmisión mediante *streaming* [10].

Formatos de almacenamiento

Son específicos para sistemas basados en *streaming*, presentan información dividida en flujos, información fragmentada para transmisión temporizada, índices de segmentos y estampas de tiempo para su reproducción, índices para saltar a diferentes puntos de la película.

Servidor

Puede funcionar bajo demanda o en directo los cuales se han explicado en el epígrafe 1.3.

Proxy

Tiene diferentes funciones según el tipo de servicio, como son: A/V bajo demanda, se almacena temporalmente la información más recientemente utilizada y la transmite a los clientes en caso de ser nuevamente solicitada; A/V en directo, reduce el número de flujos que salen del servidor Proxy [10],[19].

Red / Protocolos

Los sistemas sin control sobre la transmisión utilizan protocolos como el HTTP. Los sistemas con control sobre la transmisión poseen dos canales de comunicación entre los clientes y el servidor de *streaming.* Un canal para el control de la sesión RTSP (*Real Time streaming Protocol*) y un canal para la transmisión de la información RTP (*Real-time Transport Protocol*) /UDP (*User Datagram Protocol*)/TCP (*Transmission Control Protocol*) [10].

Cliente

Lo componen: la recepción, ya que recibe la información solicitada por el usuario. La presentación, reproduce la información recibida de forma temporizada, proporciona un interfaz para que el usuario interaccione. El buffer, se utiliza para controlar la calidad del servicio, se carga antes de comenzar la reproducción, cuando se vacía, se detiene la reproducción para recargarlo (fallo de reproducción), amortigua posibles retrasos en la llegada de paquetes (por problemas en la red o sobrecarga en el servidor). Habitualmente

existen buffers tanto de audio como de video. El tamaño del buffer debe balancear tiempo de arranque y calidad de reproducción [10],[15].

1.4.3 *Streaming* alternativo

El *streaming* alternativo se utiliza para distinguir el *streaming* tradicional de aquel que es proporcionado por un servidor web. Es decir, es el caso en el que el servidor de *streaming* no se encuentra separado del servidor web. En este tipo de *streaming* todos los pedidos de gestión son pedidos HTTP, no se encontrará en este caso que se está utilizando un determinado protocolo para controlar el *streaming* y otro protocolo para el envío de datos, como sucede en el *streaming* tradicional.

Dentro del *streaming* alternativo existen distintas opciones:

1. **Descarga trivial**: Cuando un usuario selecciona un enlace de una página que apunta a un archivo de audio o video, el *browser* sigue el mismo proceso que para un archivo de texto o imagen. Por lo tanto, el HTTP del *browser* establece primero una conexión TCP con el HTTP del servidor que aparece en el enlace. A continuación, envía una solicitud del contenido del archivo especificado en el hipervínculo mediante un mensaje de petición GET. El servidor responde enviando el contenido del archivo en un mensaje de respuesta GET. Al recibirlo, el *browser* decide a partir del campo *Content Type* de la cabecera del mensaje invocar el reproductor de video y, al mismo tiempo, le pasa el contenido del archivo comprimido. El reproductor multimedia procede a descomprimirlo y envía el flujo de *bytes* resultante a la tarjeta de sonido. Como puede evidenciarse, la desventaja de este método es que el *browser* debe recibir el contenido del archivo completo y esto puede producir un retardo inaceptable si el tamaño del archivo es considerablemente grande.

2. **Descarga Progresiva:** Se da cuando el contenido del archivo es enviado directamente al reproductor en lugar de hacerlo a través del navegador. De esta forma el reproductor procede a obtener su contenido de manera normal mediante HTTP/TCP. Al recibir el contenido del fichero, el reproductor simplemente dirige el flujo comprimido hacia la memoria de reproducción. Tras un retardo predefinido para permitir que la memoria se llene parcialmente, diez segundos en el caso del audio, comienza a leer el flujo desde la

memoria y, tras descomprimirlo, envía el flujo resultante hacia la tarjeta de audio o de video. Se realiza una descarga progresiva de la información, de manera que cuando se disponga de la información, se pueda comenzar a reproducir. Se descarga usando el máximo ancho de banda que disponen cliente y servidor, y no hay ningún control para evitar cortes en la reproducción: el medio se va almacenando en disco conforme se descarga. Si el ancho de banda es más reducido que el necesario para la reproducción, la información se reproduce "a saltos", ya que se va reproduciendo conforme llega [3].

Ventajas:

- Fácil de configurar.

Desventajas:

- El video, en su totalidad, es descargado a menos que el usuario cierre el navegador.
- El video se descarga completamente por lo que el contenido no está protegido.
- El usuario no puede reproducir el video desde cualquier momento. Solo podrá ver aquella parte que se encuentre descargada [3].

3. **HTTP *Pseudo-streaming*:** Basado en el anterior, lo que pretende es simular lo que sería el *streaming* bajo demanda, agregando la posibilidad de adelantar o retroceder la reproducción. Es decir que las partes que se saltan no se descargan y permite reducir el ancho de banda que en determinadas situaciones se pierde. *Pseudo-streaming* requiere adaptaciones tanto para el lado del cliente como para el lado del servidor. Para el lado del servidor, existen *plugins* disponibles para *Apache*, *litghttpd*. Mientras que, del lado del cliente, es necesario reproductores personalizados que permiten resincronizar el video, leer metadatos. Si se diera el caso de que se reproduzca un video que se pueda adelantar y que luego se pueda encontrar en la *cache* del *browser*, el servidor está haciendo uso de HTTP *Pseudo-streaming* [20],[21].

Ventajas:

- Posibilidad de interactuar con el *stream*
- Mejor utilización del ancho de banda

Desventajas:

- Necesita implementaciones tanto del cliente como del servidor
- El video no está protegido, es posible encontrarlo en la caché del *browser.*

4. **DASH (*Dynamic Adaptive streaming over* HTTP)**: La idea central es dividir el video en pequeñas partes y proveerlo por HTTP. Luego esas partes son combinadas en el lado del cliente y reproducidas. Este método soporta video bajo demanda y *live streaming* permitiendo entregar el *stream* correcto teniendo en cuenta cuan saturada esté la red. Existen numerosas implementaciones de este método con diferentes nombres, todas basadas en la idea antes mencionada: HTTP *Live streaming* (*Apple*), *Smooth streaming* (*Microsoft*) HTTP *Dynamic streaming* (Adobe), *Adaptive Bitrate* (*Octoshape*).

Ventajas:

- Sobre HTTP ofrece una solución muy robusta.
- Mejoras en cuanto a la protección del contenido dado que el video es dividido en pequeñas partes.

Desventajas:

- Las diferentes implementaciones no son del todo compatibles.
- Requiere un desarrollo adicional del lado del cliente [22], [23].

1.4.4 *Streaming* Tradicional vs. *Streaming* Alternativo

La primera distinción que hay que llevar a cabo a la hora de realizar la comparación es el protocolo de transporte sobre los cuales se dan cada una de las alternativas, el cual impacta notablemente en el servicio de *streaming* que se está proporcionando. El *streaming* tradicional tiene la particularidad de poder utilizarse sobre UDP mientras que el *streaming* alternativo dado que es sobre HTTP, en todas sus versiones, trabaja sobre TCP, sus principales diferencias se aprecian en la figura 1.7.

	Streaming				
	Streaming Tradicional		Streaming Alternativo		
			Progressive Download	HTTP Pseudostreaming	Dynamic Adaptive Streaming over HTTP
Protocolo de Aplicación	RTP-RTSP-RTCP		HTTP		
Protocolos de Transporte	UDP	TCP	TCP		
Protocolo de Red	IP				
Soporte Multicast	SI	No			
Soporte Unicast	Si				
Ante una perdidda de un paquete	El nivel de aplicación determina que hara con el paquete perdido	Retransmisión del mismo			
Capacidad de Multiplexación de varios Stream	SI		NO		
Problemas de NATEO y FIREWALL	Si	NO	NO		
Quality Service	SI		NO		
Permancia en cache	No		Si	SI	Solo en fragmentos de tamño reducido
Soportado por Browsers	No		SI		
	Necesario para la Reproduccción				
	Reproductor		Browser+Plugin / Reproductor		
Control sobre la transmisión	Si		No	SI	SI
Necesidad de Servidores	Servidores de Streaming		Servidores Web		
Inmediates	Se reproduce conforme llega				
	Solo usa el ancho de banda que necesita		Utilizan máximo ancho de banda para descarga		
Capacidad de Adaptación al ancho de banda	Si		NO	No	SI
Opción de modificar algortimo de compresión en curso	Si		NO		

Figura 1.7 Resumen de características de diferentes tecnologías de *streaming* [9].

TCP/Ip es usado como la capa de transporte en Internet, los archivos se bajan a la caché del *browser* tan rápido como el sistema lo permita. TCP incorpora control de flujo para manejar la tasa con la que descarga la información. No existe una tasa predeterminada para el envío de los datos. TCP incrementará dicha tasa hasta que una pérdida significativa de los paquetes que se han enviado indique que la red se encuentra congestionada. Y en dicho punto, como consecuencia, la tasa de envío disminuirá. Otra desventaja que presenta TCP para su uso en *streaming* de video, es que TCP hace uso del mecanismo de ventana deslizante para el control del flujo de datos. Esto da como resultado que los paquetes se procesen cuando llegan, por lo que si la información llega muy rápido puede darse un *buffer overflow*. En tal caso, el cliente notificará al servidor que disminuya la tasa de envío, de forma de evitar dicho inconveniente o la necesidad de memorias de reproducción grandes para ocultar el efecto de la pérdida de un segmento al usuario [9].

En el caso de que se dé una recepción incorrecta, si se utilizara UDP, la mejor opción es la interpolación de los datos o algún otro tipo de compensación, pero en este caso al estar implementado sobre TCP, se reenvía el paquete en cuestión. A su vez, en *streaming* tradicional se gana velocidad, dado que este posee un menor retardo que TCP a costa de sacrificar la confiabilidad que TCP ofrece, pero que es solucionada utilizando el protocolo RTCP y RTP. Hay que destacar también, la posibilidad del *streaming* tradicional de proveer el contenido por medio de técnicas *multicast*, ideal para la difusión de medios en vivo.

Por otro lado, con el uso de RTCP, se podrá sincronizar flujos de datos de audio y video antes de realizar la operación de descodificación. Incluso brindar la posibilidad multiservidor y la capacidad de agregar un nuevo *stream* en una presentación en vivo. Se podrá informar al emisor los paquetes que se pierden, se podrá cambiar el tipo de codificación, el tamaño del *buffer* e incluso negociar el método de transporte más adecuado antes de comenzar la transferencia del flujo de datos. Mientras que el *streaming* alternativo es una alternativa más fácil de configurar, permite dado que está implementada sobre HTTP, llegar a una mayor audiencia, y por sobre todas las cosas ofrece un servicio que combinado con un servidor web permite dar una solución acorde sin la necesidad de grandes inversiones. Dispuesto a perder determinados beneficios y la escalabilidad que el *streaming* tradicional ofrece [24].

1.5 Formatos de *streaming* de Video

Se define al formato de archivo, o *file format* por su nombre en inglés, como la estructura en la que se almacena (codifica) la información que será transmitida dentro de un archivo de ordenador. Al gestionar archivos de video se requiere una gran cantidad de datos para almacenar con precisión las señales de video, luego esta información se puede comprimir y escribir en un archivo contenedor [25].

Existen diferentes formatos de video que se puede encontrar, algunos son optimizados para la captura de video, otros para la edición y otros pocos usados para la entrega y distribución de video ya sea utilizando diferentes plataformas como Internet o un CD.

La complejidad de los sistemas de video, a diferencia de las imágenes puras, está en la coordinación que debe existir entre sus elementos: el audio, las imágenes y los metadatos.

Los metadatos, son archivos que contienen información adicional sobre el video, por ejemplo: soundtracks, idiomas, elenco, año de producción, director, subtítulos, etc. Es importante recordar que un video es mucho más que la extensión del archivo usado para almacenarlo (*.MPEG*, .MOV, etc) y puede contener un video de baja calidad o uno en alta definición 3D con audio de alta calidad.

Es usual la confusión entre los formatos de contenedores de video y los formatos de código de video. Así, se puede definir un "Contenedor de Video" como lo que se puede relacionar con el formato de archivo pues contienen los diferentes componentes del video, como son: flujos de imágenes, sonido, metadatos, etc. en esta categoría encontramos a Quick Time Mov, AVI, *MPEG*, etc [26].

Una "Señal de Audio y Video" son los datos específicos de audio y video. Un "Códec" es el software, o programa, utilizado para codificar y decodificar la señal de video o flujos. Su función es la de comprimir y descomprimir los datos para almacenarlos y transmitirlos en archivos más pequeños. En este grupo encontramos muchas opciones en el mercado pudiendo volver su elección un verdadero dolor de cabeza para el usuario.

Los Códecs pueden ser "Con Pérdidas", perdiendo parte de los datos en el proceso de compresión, pero logrando archivos de tamaño más pequeño y que corresponden a la mayoría o "Sin Pérdidas", que es lo contrario. Muchos formatos de compresión son del tipo "con pérdidas".

Normalmente los algoritmos de compresión que se emplean conllevan cierta pérdida de de datos, por lo que el objetivo es lograr la calidad más fiel al original posible produciendo un archivo lo más pequeño posible [15].

1.6 Códecs

H.261: Fue desarrollado por el grupo ITU-T y fue el primer estándar de compresión de video. Es usado principalmente en videoconferencia y videotelefonía antigua, y está optimizado por lo tanto para razones bajos de datos. Trabaja mejor en películas en las que hay poco cambio entre los cuadros. No tiene tan buena calidad como el *H.263* y puede no ejecutarse bien en máquinas de gama más baja [27].

H.263: Inicialmente creado para videoconferencia y video por internet, este códec fue un gran paso hacia la estandarización de la capacidad de compresión de video de escaneo progresivo, fue usado principalmente como punto de partida para el desarrollo de *MPEG* (que está optimizado para razones de datos más elevadas). En la actualidad es usado para comprimir video en formato Flash. Entre sus desventajas está que hace un uso bastante intensivo del CPU y puede no dar buenos resultados en máquinas de gama más baja [27].

Logra imágenes de buena calidad tanto a diferentes razones de bits y una mejor calidad de imagen que *MPEG-2*, *MPEG-4* o *H.263*. Es dos veces más eficiente que *MPEG*-4. Es fácil de integrar y cubre un amplio rango de formato de imágenes. Entre sus desventajas encontramos que requiere un tiempo de codificación mayor y que los acuerdos de licencia son algo complicados.

MPEG-1: Este logra buena calidad de imagen a razones relativas de CD-ROM, se usa principalmente en VCD, o video CD. *MPEG* incluye compresión de video y audio. El mayor problema que posee son sus altos requerimientos para la reproducción siendo esta su principal desventaja.

MPEG-2: Está optimizado para calidad de difusión para video digital y ofrece muy buena calidad de imagen y resolución. Es el estandar de vídeo principal para DVD-Video. Se requiere pagar la licencia para distribuir video con *MPEG-2* [27].

DiVx: Codec creado por *DiVx* Inc es un códec comercial, que no es libre de costo. Este códec utiliza compresión *lossy* (con pérdida) *MPEG-4 Part 2* y es totalmente compatible con *MPEG-4-Advanced Simple Profile* (*MPEG-4* ASP). Es bastante simple de utilizar y es popular debido a su facilidad para comprimir largos segmentos de video en tamaños pequeños manteniendo una calidad visual relativamente alta [27].

Existen muchos tipos de Códecs, siendo los más versátiles los de la familia H.264, también llamado *MPEG-4 Part* 10 o *AVC*. Este códec provee alta calidad de codificación y decodificación para aplicaciones de transmisión de video en tiempo real, a razones que van desde un cuarto a la mitad del tamaño de los archivos de los formatos de video previos. El tamaño de archivo logrado es 3 veces más pequeño que los logrados con los códecs *MPEG-*

2, soporta video 3-D y varios esquemas de codificación de audio. Al usarse con una alta tasa de bits provee una calidad del nivel de un disco *Blue Ray*, aunque también es bastante útil cuando se prioriza la compresión en la transmisión además de ser soportado por múltiples plataformas y servicios como Vimeo y Youtube [25].

x264: Es una implementación abierta y disponible de forma open source del estándar H.264. Ofrece buena calidad y compresión de archivo se utiliza con frecuencia en grabadoras de vídeo AVCHD y en reproductores de HD DVD y Blu-ray. En marzo de 2012, en forma conjunta varias comunidades de uploaders decidieron utilizar este códec y dejar de usar Xvid/avi, por la mayor calidad y compresión a resoluciones SD [27].

En la actualidad hay varios códecs que gestionan videos de Alta Resolución 4K (con resolución de 3840x2160), Ultra HD o UHD, que van más allá de la transmisión a 1080p (*Full High Definition* con resolución de 1920x1080). Los dos principales, cuya finalidad es muy parecida y considerado sus características de calidad, tiempo de codificación, requerimientos de CPU tanto en plataformas Windows como MAC, son el HEVC puesto en marcha por el *MPEG* e incluido por Apple en sus dispositivos a partir de iPhone 6 y VP9 de Google [28].

1.7 *Streaming* propuesto por YouTube y Netflix

El *streaming* propuesto por YouTube funciona usando la tecnología de Adobe. El video llega incrustado como parte de un archivo cuya extensión es swf. Entonces, cuando se desea ver un video en Youtube, lo que está sucediendo es que este devuelve un .swf que se descarga a la caché del *browser* y luego es ejecutado por *Adobe Flash Player*. De esta forma, este .swf se encarga de embeber el video en el *browser*, chequear como se ve el reproductor y controlar el comportamiento del video. Siendo el parámetro *video_id*, el que indica el id del video a reproducir. A la hora de reproducir un video no se hace uso del protocolo RTSP. Todas las peticiones que se realizan para interactuar con el video, particularmente el reproducir y el adelantar se llevan a cabo mediante *http_request*.

Se puede concluir que Youtube no muestra sus videos mediante *streaming* tradicional ni tampoco HTTP Dynamic *streaming*. Youtube lleva a cabo pseudo-*streaming* para la

entrega y reproducción de sus videos. No tiene en cuenta el ancho de banda del cliente y tampoco envía los videos al *bitrate* que son codificados, usa en todos los casos la máxima velocidad de descarga. Los videos quedan alojados en la caché del *browser* y todo su mecanismo de *streaming* se basa en HTTP, incluso las acciones de play y adelantar. En todos los casos hace uso de TCP [29].

NetFlix provee su contenido multimedia, bajo la tecnología de *streaming* denominada *Silverlight*. *Silverlight*, es un producto desarrollado por Microsoft, que permite crear aplicaciones web, particularmente RIA (*Rich Internet Applicaction*). A pesar de que *Silverlight* tiene infinidad de utilidades se concentrará la atención en la tecnología utilizada para llevar a cabo el *streaming* que es *Smooth streaming*. Esta tecnología no es otra cosa que DASH, y es lo que realmente utiliza Netflix a la hora de proveer un video. De esta forma, Netflix posee sus películas codificadas en varias calidades, y detectando el ancho de banda, descargará aquel que se adecue a la velocidad negociada. Esto implica que ver un video con Smooth *streaming* sea mucho más suave y rápido que con otros métodos de *streaming* tradicionales. Al comenzar a ver una película se descarga un archivo .xap, que determinará el funcionamiento del reproductor. Si se inspecciona ese .xap, no es otra cosa que un archivo comprimido que posee todo lo necesario para poder correr una aplicación con el plugin *Silverlight*.

Al descompactar el archivo se encuentran como mínimo dos archivos, un *manifest* y el archivo propio que contiene el código necesario para la ejecución del reproductor (.dll). Sin embargo, lo que realmente sucede con Netflix es que el .xap suele guardarse en la cache del *browser*, con un archivo .dll y luego, se baja el *manifest* por separado. Este *manifest*, describe los diferentes códecs que posee la película tanto para el audio como para el video, identificándolos con un Id y la url (*Uniform Resource Locator*) del host donde se encuentran alojados.

De esta forma, cuando un cliente solicita un fragmento en determinada ubicación, el servidor encuentra de forma dinámica el *Movie Fragment box* con el *MPG-4* contiguo y solo envía este a través de la red. En otras palabras, el servidor no tiene una película dividida en millones de fragmentos, sino que estos son creados dinámicamente por cada solicitud. Lo que da lugar a ahorros en lo que respecta a manejo de archivos. Todo esto da

lugar a que el video se vaya descargando en pequeñas partes y no en una sola unidad como en la descarga progresiva, que es el caso de YouTube [30].

1.8 Plataformas para emitir *streaming* de video

VideoLAN

Es una completa solución de *software* para el *streaming* de video, desarrollada en comunidad por todo el mundo, bajo la licencia pública general GNU (GPL). Está diseñado para generar *stream* de videos *MPEG* sobre redes de trabajo de altos anchos de banda.

La solución de VideoLAN la componen dos herramientas, VLS (VideoLAN Server) y VLC (inicialmente el cliente VideoLAN). El programa se puede invocar por línea de comandos o por interfaz gráfica (GUI). Se prefiere el uso de la primera en el lado del servidor, ya que permite un mejor entendimiento de los parámetros de puesta en marcha [31].

DaCast

Es una plataforma de video en línea que permite a las empresas transmitir contenido de video y ofrecer una programación gratuita o de pago.

DaCast es una plataforma de video en línea y de alojamiento de video de autoservicio. Su sistema permite a los usuarios controlar sus transmisiones en vivo y videos a pedido. DaCast se posiciona como una empresa SaaS (software como servicio) con el eslogan siguiente: "*streaming* como Servicio".

La plataforma permite a los usuarios monetizar su contenido de video vía un integrado *paywall* en el reproductor multimedia o vía integración de anuncios.

La distribución de contenido de audio y video es basada en la tecnología HTML5. El protocolo de la transmisión en vivo es RTMP y se soporta la entrega de formatos HLS y HDS.

La plataforma es compatible con formatos de archivo de video diferentes como .MOV, .MP4, Mp3, .M4Un o AAC y se puede transcodificar otros formatos para hacerles compatible con el servicio [32].

Class on live

Es una herramienta muy buena si lo que se desea es crear, gestionar, promocionar e impartir cursos *online*, clases, talleres, webinars, etc, es decir, está totalmente enfocada a la formación en *streaming*.

Con un diseño muy sencillo e intuitivo, también te permite vincularlo con tu web con un widget de reservas que funciona a través de un código que te proporciona la plataforma.

Algunas de sus funcionalidades más destacadas son:

- Se puede impartir clases por videoconferencia y en *streaming* sin necesidad de conocimientos técnicos.
- Funcionalidades extras como el envío de emails automáticos o la gestión de los pagos [32].

***MPEG*4Ip**

Provee un sistema de fin a fin para explorar el mundo del *streaming*. El paquete incluye muchos sistemas de código abierto y los junta para utilizarlos integrados en un solo paquete. Esta es una herramienta para el *streaming* de video y audio que es basada en estándares y es libre de protocolos y extensiones propietarios.

Una de las herramientas que contempla *MPEG*4Ip, es la aplicación MP4Live, la cual puede producir flujos de audio y video para ser transmitidos por la red utilizando el protocolo RTP en conexiones *unicast* o *multicast*.

MP4Live dispone de una interfaz gráfica que permite configurar los parámetros necesarios para generar el *streaming*. La configuración puede ser también leída desde un archivo llamado .mp4live_rc, el que es almacenado en el directorio *home* del usuario.

Una vez seteados los parámetros de audio y video en el servidor, es posible generar el flujo para transmitirlo o almacenarlo en un archivo mp4.

Por el lado del cliente, este debe reconocer los *códecs* utilizados. Por lo general, se transmite en *MPEG-4* con el *códec* de video XviD y el audio en AAC o MP3. *MPEG*4Ip dispone también de su *player* llamado MP4PLAYER, que puede ser compilado para varias plataformas.

Para un mejor control de los flujos generados por MP4LIVE, los creadores de *MPEG*4Ip recomiendan el uso del servidor (DSS) *Darwin streaming Server* [32].

Dynebolic

No es una aplicación de *streaming* en sí, sino que corresponde a una distribución en *Live* CD, la cual entrega un entorno de producción multimedia bastante práctico. Cuenta con herramientas para capturar, manipular, codificar y transmitir audio y video. Tiene soporte para una gran cantidad de *software* libre y no es necesario instalar nada en la computadora.

Algunos de los programas para generar *streaming* de video que provee son MP4LIVE y PALANTIR. Distribuciones como estas son útiles cuando no se dispone de un equipo dedicado dispuesto como servidor de medios, o bien cuando se desea transmitir ocasionalmente [33].

1.9 Conclusiones

A lo largo de este capítulo, se caracterizó el *streaming* de video, se expusieron varios conceptos hasta llegar al más abarcador. Se definieron los servicios que este ofrece, también se explicaron los elementos básicos de una arquitectura clásica de *streaming*, y se compararon las diferentes tecnologías con las que se puede implementar.

.

CAPÍTULO 2. Servicio *Streaming* en la red UCLV /Implementación del servicio *streaming* con el uso de *FFmpeg*.

2.1 Introducción

En este capítulo se explicará como será el servicio *streaming* que se desea. Se hará una breve descripción del proceso de captura de video en GNU/Linux y su posterior transmisión sobre redes Ip y se expondrán cómo ajustar las principales herramientas de usadas para el diseño del servicio *streaming* en la uclv.

2.2 Servicio *streaming*

En este trabajo se implementa un servicio *streaming* de video en tiempo real donde se transmitan los eventos que están sucediendo justo en el momento de la difusión, independientemente de cuando se conecta un cliente al servidor, todos ven exactamente el mismo punto del *stream* en un instante determinado (excepto las lógicas variaciones de los retardos en la red que hacen que unos clientes reciban antes los datos que otros). Para poder efectuar este tipo de transmisión no es suficiente con disponer de un servidor de *streaming*, sino que también es necesario un equipo que realice el proceso de captura y compresión en tiempo real (que a veces se conoce como difusor o *broadcaster*), este servicio de *streaming* será con información en vivo.

Para desarrollar un sistema de transmisión de video con las características que se requieren, se necesita desarrollar una infraestructura como la que se presenta en la figura 2.1. La red sobre la cual descansa, debe soportar altas tasas de transferencia. Básicamente, la velocidad de transmisión está íntimamente ligada a la calidad de la imagen y los códec utilizados [33].

Figura 2.1 Esquema de un sistema de *streaming*[15].

Todo el *software* utilizado corresponde en su gran mayoria a la categoría de *Software* Libre, en sus distintos tipos de licencias. Como consecuencia de lo anterior se destaca que, por un lado, los costos de implementación son muy bajos (uno de los factores más importantes) y por otro lado que existen numerosos sitios de Internet donde se puede buscar información sobre ellos.

A nivel conceptual, la transmisión del *streaming* involucra diversas etapas como se puede ver en la figura 2.2.

Figura 2.2 Etapas conceptuales del proceso de *streaming* [33].

2.3 Envío de video al servidor

Como el servidor se encuentra en un lugar remoto y las grabaciones en vivo se realizaran desde cualquier punto de la universidad con la única necesidad de estar cerca de un punto con acceso a la red universitaria para la el envío del video al servidor. El *streaming* de video se podrá realizar desde una webcam, cámara Ip y hasta de un movil

2.3.1 Por Webcam

Para enviar el *streaming* de video desde la webcam de la Pc se puede usar ***FFmpeg*** donde la línea de codigo en el terminal es:

FFmpeg *-y -f video4linux2 -i /dev/video0 -f alsa -i hw:0,0 -f MPEGts -flush_packets 0 UDP://dirección Ip del servidor:1234?pkt_size=131*

Donde:

-y	*sobrescribe archivo de salida sin pedir confirmación*
-f	*llamadas a servicios de audio y video externos a* ***FFmpeg***
-i	*ubicación física de los servicios de audio y video*
-flush_packets	*escribe los paquetes inmediatamente*
?pkt_size=131	

Al emitirse *MPEG*-TS a UDP y usarse la opción pkt_size, ***FFmpeg*** no produce una salida con un tamaño de paquete constante; en su lugar, la opción se utiliza como límite del tamaño máximo de paquete. Al mirar las fuentes, de hecho, es un tamaño máximo de paquete.

2.3.2 Por cámara Ip

Una Cámara Ip (también conocida como cámaras Web o de Red) es una videocámara especialmente diseñadas para enviar las señales (video, y en algunos casos audio) a través de Internet desde un explorador (por ejemplo el Internet Explorer) o a través de concentrador (un HUB o un SWITCH) en una Red Local (LAN) [34].

Inicialmente se debe conectar la Cámara Ip al computador, directamente usándose el cable de red cruzado, como se muestra en la figura 2.3. Posteriormente se enciende tanto la computadoracomo la Cámara Ip conectándola a la alimentación de energía, normalmente todas incluyen en la caja un adaptador de energía (No debe utilizarse un adaptador de energía diferente al original, ni extender la distancia del mismo realizando empalmes).

Figura 2.3 Conexión de cámara Ip a la PC usando un cable de red cruzado [35].

Una vez conectados los equipos, se pasa a identificar la dirección Ip por defecto de la Cámara, para que una vez ya conocida ésta cambiarla (Es muy importante que tanto la Cámara Ip como la computadorase encuentren en la misma subred)

Para verificarse la dirección Ip por defecto o de fábrica de la Cámara, se puede consultar ésta en el manual del equipo o contactando al proveedor de la misma. Por lo general todas las marcas de Cámaras Ip tienen un procedimiento de configuración muy rápido, casi que *Plug And Play*, se hace uso de una herramienta muy común en todas las marcas denominado Ip Utility. El Ip *Utility* es un aplicativo que por lo general viene en el cd de instalación que acompaña la cámara Ip, y permite de una manera muy fácil encontrar la dirección Ip y referencia de la cámara ya conectada en una red.
Una vez instalada la camara Ip según diga su manual e instalado el softwere de la cámara en la Pc se realiza un ping desde la computadorahacia la Ip de la cámara. Si se recibe respuesta de ping de la cámara, ya se puede acceder a ella para ver su video en vivo utilizando un navegador web compatible. Es importante chequear el navegador web compatible con la cámara Ip que se utiliza, por lo general la compatibilidad con navegador web la encontramos en la parte exterior de la caja de la Cámara o internamente en su manual [35].

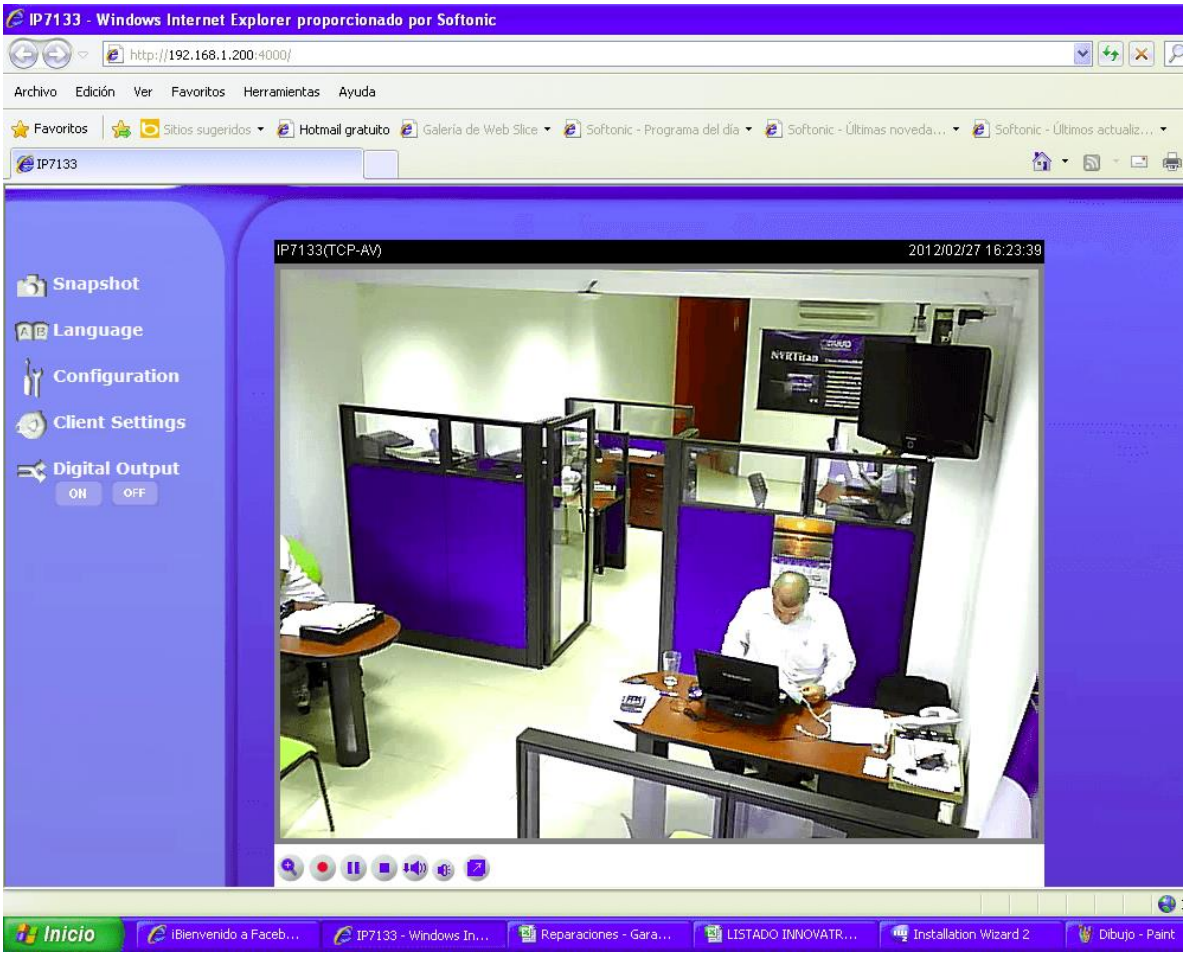


Figura 2.4 Visión de la cámara Ip dese el navegador.

Se puede acceder a la grabación poniendo su dirección Ip y el puerto de salida en el navegador como se muestra en la figura 2.4 o accediendo a ella desde un reproductor de video como el VLC. En caso de accederse a la grabación por ***FFmpeg*** desde el terminal se realiza de la siguiente línea de comandos

.FFmpeg *-i http://dirección Ip de cámara_Ip :puerto/ http://dirección Ip del servidor:puerto/*

2.3.3 Por teléfono móvil

Desde los teléfonos móviles también se puede hacer grabaciones para la realización del video *streaming* en tiempo real.

El teléfono funcionará como una webcam, como se aprecia en la figura 2.5 y solo se necesitará una portátil con Windows aunque también hay una pequeña guía para GNU/Linux, un móvil Android y una buena conexión WiFi [36],[37].

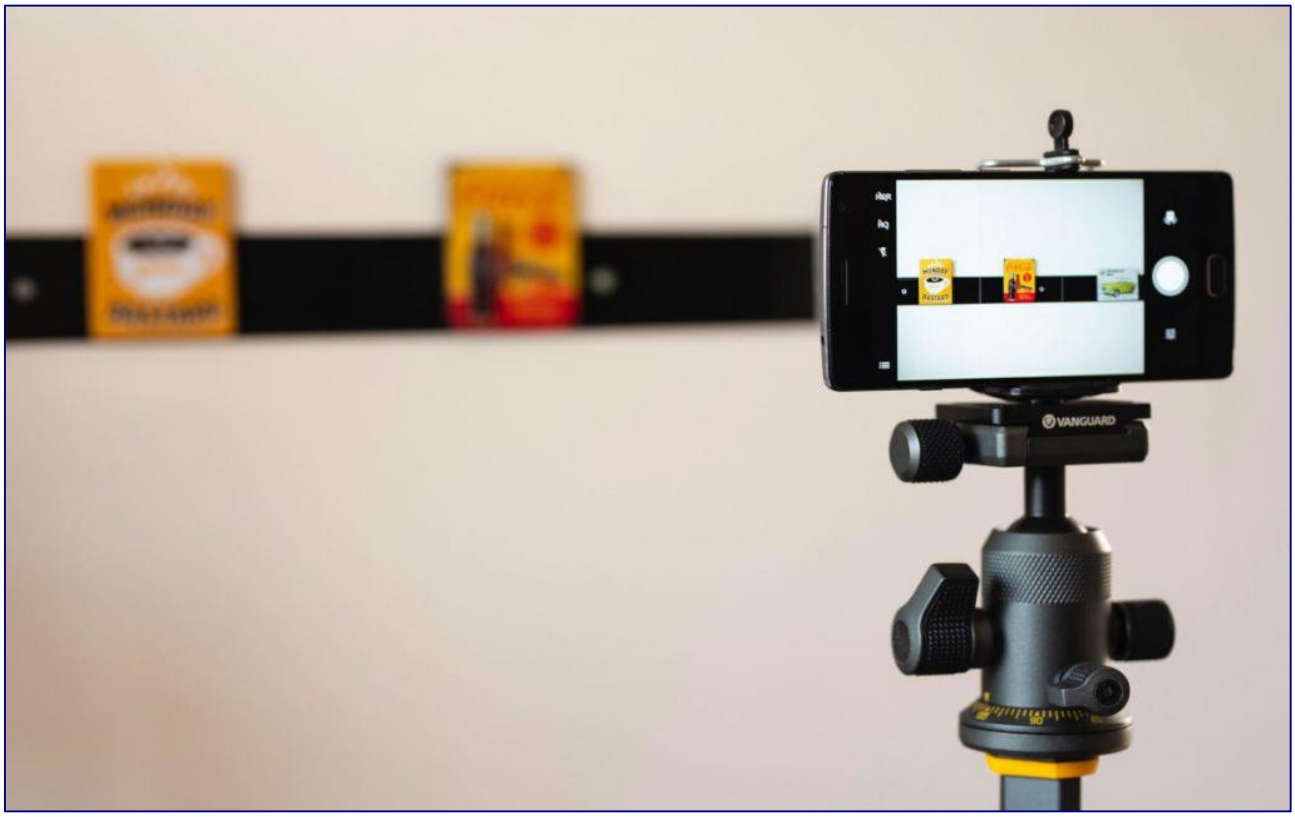


Figura 2.5 Móvil como webcam [38].

Para convertir el celular android en una *webcam*, se necesita instalar una aplicación en el móvil, y un *script* en Ubuntu. La primera parte de esta combinación, es la instalación de la aplicación que convertirá el móvil en una cámara Ip , lo que permitirá utilizar el movil Android como webcam. Se trata de Ip Webcam (disponible en https://play.google.com/store/apps/details?id=com.pas.webcam). Esta aplicación, permite ver la cámara en cualquier sistema operativo, pudiéndose utilizar un navegador o incluso el reproductor VLC, Skype, etc. También se puede ver la cámara desde otro dispositivo Android [38].

Esta aplicación es totalmente compatible con Windows, Mac y Linux, lo cual nos ofrece un amplio abanico de posibilidades y de distintos usuarios finales, además su funcionamiento es tan sencillo como copiar la dirección de Ip generada y conectarse a la cámara de manera inalámbrica, gracias a la conexión wifi de los dispositivos Android [39], [40].

Una vez instalada *Ip Webcam*, se selecciona y se accede a la aplicación. La misma contiene todo un conjunto de opciones que permiten configurar la aplicación hasta el más mínimo detalle.

Para empezar a utilizar el móvil Android como webcam solo hay que utilizar la última de las opciones. La que dice iniciar servidor. Al hacerlo, en la parte inferior se indica la dirección a la que se debe dirigir el navegador para ver lo que muestra tu Android como webcam. La dirección podría ser http://192.168.1.252:8080. Al abrir Firefox, aparece un menú con diferentes entradas. Primero se abre la pestaña “Inicio”. Para poder ver lo que se está grabando y se elige el renderizador de vídeo que se quiera. Es mejor usar Navegador o JavascrIpt. También se puede utilizar la opción de “Pantalla completa”, aunque esto lo único que hace es abrir el renderizado de vídeo en una pestaña nueva.

Respecto al audio también tiene unas opciones parecidas al vídeo. Aunque es recomendable utilizar cualquiera de las opciones “HTML”, ya sea “HTML Wav” o “HTML Opus”. Además, en el caso del audio tiene una opción adicional, que es la de poder utilizar audio bidireccional. De esta manera no solo puedes escuchar lo que se está oyendo en el lado del móvil Android como webcam, sino que también se podrá escuchar desde el lado de la Pc.

En la parte inferior de la pestaña donde se ve lo que se está grabando, se puede acceder a todos los controles [38].

En esta aplicación se encontrarán las siguientes opciones:

- Un control de grabador, donde se puede poner una etiqueta para el caso de que se grabe lo que se está viendo.
- Se puede seleccionar el tipo de grabación. O bien una grabación manual o bien circular. Una grabación circular, lo que significa es hacer grabaciones de una

duración determinada. Cuando termina esa grabación empieza una nueva. Y en caso de que no haya espacio en el disco, se comenzarán a sobrescribir las grabaciones antiguas.

- Tomar una foto.
- Hacer una foto y guardarla en el móvil.
- Tomar una foto, pero esta vez realizando un autoenfoque.
- Y lo mismo, pero guardándose en el móvil.
- Realizar zoom.
- Recortar la imagen que estás viendo.
- Modificar la calidad de emisión.
- Otras opciones variadas son el mantener el autoenfoque, encender el flash o la visión nocturna.
- Elegir entre la cámara frontal o trasera.
- Configurar la detección de movimiento y ajustar la sensibilidad.

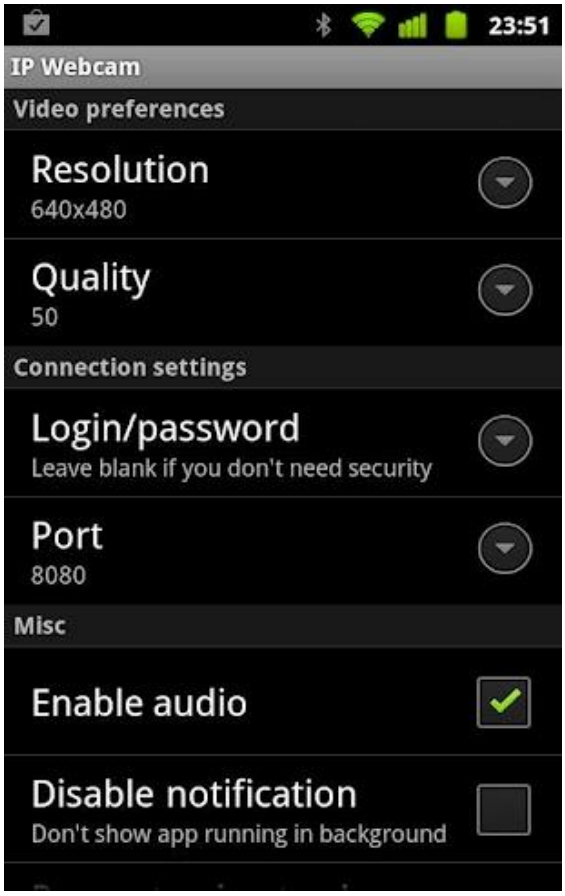


Figura 2.6 Configuracion de Ip Webcam [41].

Además, se puede hacer una configuración más avanzada, como la mostrada en la figura 2.6, donde se podrá:

- Definir áreas de detección de movimiento
- Establecer el modo de enfoque
- Modificar la ganancia de visión nocturna
- Cambiar la exposición de visión nocturna
- Establecer la resolución de vídeo y de fotos.
- Cambiar la orientación, poner la imagen en espejo o incluso voltear.

- Aplicar efectos de color, tipo solarización, sepia, posterización, agua o pizarrón blanco.

Además de esta configuración, Ip Webcam, viene con diferentes scripts que se pueden activar para realizar operaciones adicionales. Algunos de estos scripts son los siguientes:

- Activación automática de la visión nocturna a determinadas horas, según defina el usuario.
- Empezar a grabar cuando se inicie la aplicación.
- Enviar fotos por correo electrónico al detectar un movimiento
- Ocultar la pantalla
- Guardar una foto si se detecta movimiento
- Realizar fotos de forma regular.
- Subir las fotos y vídeos a un servidor remoto.

Además, es posible utilizar desde alguna aplicación como puede ser por ejemplo Skype o VLC. En este caso, es necesario que se utilice los controladores Linux. El controlador de Linux permite usar tanto el audio como el vídeo en Ubuntu en, aunque también es posible utilizarlo en otras distribuciones. El controlador es gratuito y de código abierto. Está disponible para distribuciones como Debian, Ubuntu, Linux Mint y Arch.

Se trata de un script, que permite utilizar el móvil Android como webcam o como un micrófono en Linux. De esta manera Ip WebCam, funciona como un servidor en el móvil, que sirve *MPEG* como vídeo y WAV como audio, mediante el protocolo HTTP a través del puerto 8080. Es posible modificar el puerto desde la configuración del móvil.

Para su instalación se tiene que descargar el script Ipwebcam-gst (disponible en https://github.com/bluezio/Ipwebcam-gstb)desde GitHub.

Una vez descargado se debe que modificar “WIFI_Ip” por la Ip del móvil, y el puerto “PORT” por el que se haya definido la aplicación. También se puede modificar las dimensiones del vídeo [39].

Una vez modificado estas variables, se tendrá que ejecutar el script cada vez que se quiera utilizar el móvil en Ubuntu. Para ello, se ejecuta la orden:

prepare-videochat.sh

Esto es un poco tedioso, porque cada vez que se realice la conexión pedirá las credenciales de administrador.

Otra apk para este fin es DroidCam. Su versión gratuita es más que suficiente para un uso puntual, pero si quieres conseguir resolución HD y eliminar los anuncios de la app se tiene que buscar la versión Pro.

Debes descargar la aplicación en tu móvil Android desde Google Play. Una vez instalada, accede a la web oficial de Dev47Apps y descarga e instala el cliente de escritorio. Una vez funcionando tu ordenador entenderá que la imagen viene de la webcam y podrás emitir tantos videos como quieras [41], [42].

Para enviar la grabación del móvil desde la PC por el terminal hacia el servidor de *streaming* es de forma semejanete a la descrita en el epígrafe anterior pero la direccion Ip y puerto de la entrada de video sería la del movil.

2.3.4 Por VLC

VLC media player es un reproductor y framework multimedia, libre y de código abierto desarrollado por el proyecto VideoLAN. Es un programa multiplataforma con versiones disponibles para muchos sistemas operativos, es capaz de reproducir casi cualquier formato de vídeo sin necesidad de instalar códecs externos y puede reproducir vídeos en formatos DVD, Bluray, a resoluciones normales, en alta definición o incluso en ultra alta definición o 4K.

En VLC tambien está la opción de emitir un video desde la webcam, desde la red o desde otro dispisitivo de captura se puede emitir con transcodificación o sin ella. Emitir un video es muy sencillo para ello los pasos son los siguientes:

Ir a “Medio” en el seleccionar “Emitir” (Ctrl+S)

Donde se puede emitir video desde un archivo ya existente en la Pc o en un disco hasta un flujo de video proveniente de una direccion URL y tambien desde la webcam de la propia Pc.

Una vez seleccionado el video o flujo que se quiere emitir pulsamos emitir

Posteriormente seleccionamos hacia donde emitiremos la grabación donde contamos con varias opciones como:

- Archivo
- HTTP
- MMSH
- RTSP
- RTP
- UDP
- Icecast

Al añadir el protocolo de envío y el destino se elige si se quiere o no transcodificar y en que formato se realizará, como se muestra en la figura 2.7.

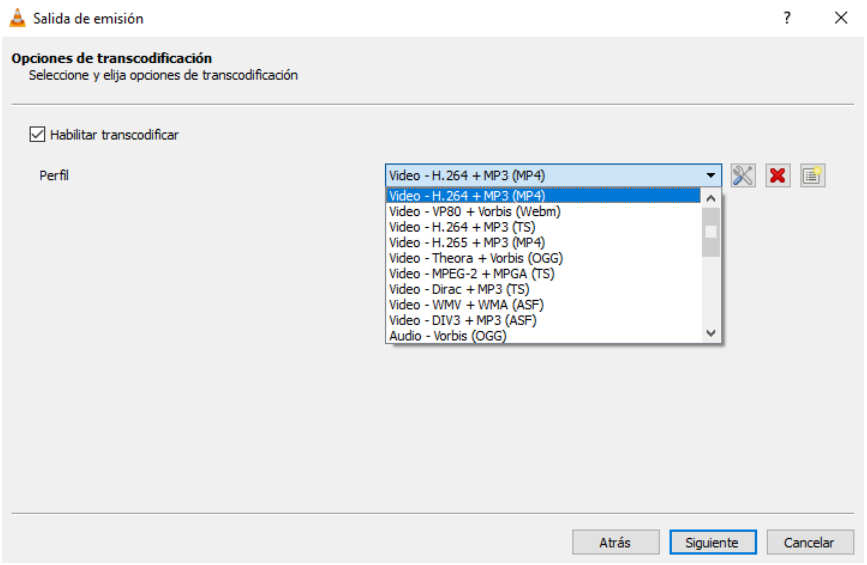


Figura 2.7 Opciones de transcodificación.

Una vez cumplido estos pasos el reproductor iniciará la emisión inmediatamente hacia el destino seleccionado.

2.4 *FFmpeg*

FFmpeg es una colección de software libre que puede convertir (transcodificar) audio y vídeo. ***FFmpeg*** está desarrollado en GNU/Linux, pero puede ser compilado en la mayoría de los sistemas operativos, incluyendo Windows. Desde el abandono del proyecto *MEncoder*, ***FFmpeg*** se ha convertido en la mejor opción para codificación audio-vídeo en GNU/Linux, una auténtica "navaja suiza de la codificación". ***FFmpeg*** es un programa bastante sencillo y muy fácil de usar, orientado tanto a personas con conocimientos avanzados como usuarios novatos. Es capaz de elegir el códec con sólo escribir la extensión. Por ejemplo, ***FFmpeg*** usará x264 si se elige .mp4, *MPEG*4 si se usa .avi, VP8 si se usa .webm, etc [43].

Es una herramienta de alta rapidez en línea de comandos que facilita la conversión de archivos de audio y/o video de un formato a otro, capturar y codificar en tiempo real a partir de distintas fuentes de captura como por ejemplo una webcam o una tarjeta de TV, o realiza un *screencast*. Esta herramienta ofrece un manejo eficiente de los parámetros característicos de audio y video como imágenes por segundo (fps), resolución, relación de aspecto, tasa de bits, compresión, número de canales de audio, entre otros. Esta plataforma fue la primera que contó con las librerías correspondientes a HEVC/H.265, lo que la califica como una de las pioneras en el uso de este estándar de codificación. ***FFmpeg*** es una solución completa de código libre y de plataforma cruzada. Es un *software* libre licenciado bajo LGPL o GPL (*General Public License*) en dependencia de la elección en las opciones de configuración [43]. Los princIpales elementos que conforman la plataforma ***FFmpeg*** son:

FFmpeg: herramienta que permite el procesamiento de archivos multimedia y su conversión hacia diferentes formatos compatibles. Además, puede capturar y codificar en tiempo real desde *DirectShow*, una tarjeta de televisión u otro dispositivo compatible.

Libavcodec: biblioteca que contiene todos los codecs de ***FFmpeg***, para consultar la información acerca del *streaming*, y para codificar/decodificar el archivo multimedia. Aunque el objetivo princIpal de esta biblioteca son los archivos de audio/video, también provee codecs para formatos de imágenes como GIF (Graphics Interchange Format**)** y PNG (Portable Network Graphics). Además, muchos codecs fueron desarrollados desde cero para asegurar una mayor eficiencia y la reutilización de los mismos.

Libavformat: biblioteca que contiene los multIplexores y demultIplexores para determinar el tipo de archivo, e identificar sus codecs asociados.

FFserver: trabaja en conjunto con ***FFmpeg*** puesto que funciona como un servidor *streaming* de audio y video de todos los ficheros que ***FFmpeg*** pueda usar y procesar. Soporta HTTP (Hypertext Transfer Protocol).

FFplay: reproductor de archivos multimedia, con características simples y de carácter portable.

FFprobe: herramienta que obtiene e imprime en pantalla la información de los recursos multimedia previamente procesados. Las bibliotecas que se utilizan son las siguientes:

Libavutil: biblioteca de apoyo que contiene todas las rutinas comunes en las diferentes partes de ***FFmpeg***.

Libpostproc: biblioteca de funciones de post-procesado de video.

Libswscale: biblioteca de escalado de video [44].

Estructura ***FFmpeg***

FFmpeg *[options] [[infile options] -i infile]... {[outfile options] outfile}…*

2.5 *FFserver*

FFserver es un servidor de transmisión de audio y video. Admite varias fuentes en vivo, transmisión de archivos y cambio de hora en las fuentes en vivo. Puede buscar posiciones en el pasado en cada transmisión en vivo, siempre que especifique un almacenamiento de alimentación suficientemente grande.

FFserver se configura a través de un archivo de configuración, que se lee en el inicio. Si no se especifica explícitamente, se leerá desde */etc/**FFserver**.conf*.

FFserver recibe archivos pregrabados o transmisiones FFM de alguna instancia de ***FFmpeg*** como entrada, luego las transmite sobre RTP / RTSP / HTTP.

Una instancia de ***FFserver*** escuchará en algún puerto como se especifica en el archivo de configuración. Puede iniciar una o más instancias de ***FFmpeg*** y enviar una o más transmisiones de FFM al puerto donde ***FFserver*** espera recibirlas. Alternativamente, puede hacer que ***FFserver*** ejecute tales instancias de ***FFmpeg*** durante el inicio.

Los flujos de entrada se denominan feeds y cada uno se especifica mediante una sección "<Feed>" en el archivo de configuración [45].

Un feed es un flujo de FFM creado por ***FFmpeg*** y enviado a un puerto donde ***FFserver*** está escuchando.

Cada fuente se identifica con un nombre único, que corresponde al nombre del recurso publicado en ***FFserver*** y se configura mediante una sección dedicada de "Fuente" en el archivo de configuración.

La URL de publicación del feed viene dada por:

*http: // <**FFserver**_direccion_Ip >: <Puerto_http> / <nombre_feed>*

Para cada fuente puede tener diferentes flujos de salida en varios formatos, cada uno especificado por una sección "<Stream>" en el archivo de configuración [45].

La URL HTTP de acceso al flujo está dada por:

*http: // <**FFserver**_Ip_address >: <http_port> / <stream_name> [<options>]*

FFserver se inicia:

***FFserver** -d -f /dirección donde se encuentra el archivo .conf*

2.6 Codificación con H.265

El formato de compresión de video H.265 o *MPEG*-H Parte2, también llamado de forma común como HEVC, fue desarrollado en conjunto por *VCEG (Video Coding Experts Group)* y *MPEG (Moving Picture Experts Group)* [46] [47].

Se estableció como el sucesor del H.264/*MPEG-4* (*AVC*) en el 2013. HEVC/H.265 mejora las herramientas existentes en los estándares previos, específicamente en H.264, logrando una mejora en la calidad percibida y permitiendo una reducción de *bitrate* por encima del 50% con respecto a H.264. H.265 es altamente eficiente en dos sentidos: lograr eficiencia en formatos de muy alta resolución como UHDTV (*Ultra High Definition Television*), y hacerlo también en entornos de muy baja tasa binaria, como es el caso de servicios *streaming*, WebTV y OTT (*Over the Top Content*). H.265, además, utiliza la estimación y la compensación de movimiento para aprovechar la similitud temporal de las secuencias, y además emplea la transformada discreta del coseno, con cierta variación, con el objetivo de aplicar la cuantificación en el dominio transformado, y codificar estadísticamente dichos coeficientes. Para ello H.265 cuenta con diferentes estructuras novedosas que se analizarán a continuación y se muestran en la figura 2.8 [44], [48].

- Nueva Unidad de Codificación: Define una nueva estructura de codificación que diverge de las anteriores unidades denominadas macro bloques con dimensiones de 16x16 píxeles. En este estándar la unidad de codificación se denomina *Coding Tree Block* (CTB) y sus dimensiones pueden ser desde 8x8 hasta 64x64 píxeles. En HEVC, una imagen se divide en diferentes CTU (*Coding Tree Units*) que pueden presentar un tamaño máximo de LxL píxeles, para el caso de la luminancia L puede tomar los valores de 16, 32 ó 64, con los tamaños más grandes que permiten típicamente una mejor compresión.
- Unidad de Predicción: H.265 define una nueva unidad de predicción (PU) que puede tener dimensiones desde 32x32 hasta 4x4, lo que representa la mitad que la menor de las unidades de codificación. Al igual que su antecesor H.264, pueden aplicarse de modo intra-frame o inter-frame, utilizando divisiones distintas.

- Predicción Intra-Frame: HEVC introduce un nuevo método de predicción intra-frame. El método intra-angular define 33 predicciones direccionales para todos sus posibles tamaños de PU el cual se diferencia de los 2 predictores direccionales que define H.264 para tamaños de 16x16, y 8 predictores para 8x8 y 4x4. Además, se definen los modos intra-planar, donde se supone una superficie de amplitud con una pendiente horizontal y vertical derivada de los límites, y el modo intra-DC donde se asume una superficie plana con un valor que coincida con el valor medio de las muestras de contorno.

- Predicción Inter-Frame: La estimación de movimiento en H.265 se lleva a cabo con la precisión de ¼ de píxel con filtros interpoladores de 7-taps y 8-taps. Se incorpora además la AMVP (*Advanced Motion Vector Prediction*) para lograr la señalización de sus vectores de movimiento mediante el cálculo de los vectores de movimiento más probables obtenidos desde los bloques vecinos.

- Unidades de Transformación: Se define una nueva TU (*Transform Unit*). Se aplica de igual forma la DCT (*Discrete Cosine Transforme*) con una ligera modificación con respecto a la utilizada en H.264, también permite bajo ciertas condiciones la utilización de la DST (*Discrete Sine Transforme*). En el caso de H.265 se amplían los tamaños de transformación desde 4x4 a 32x32.

- *Sample Adaptive Offset* (Compensación Adaptable de la muestra): Nueva herramienta aplicada luego del *In-loop filter* con el objetivo de lograr mejorar la calidad subjetiva de la imagen decodificada, aplicando pequeños *offset* a los píxeles decodificados, en función de la zona a la que pertenecen [48].

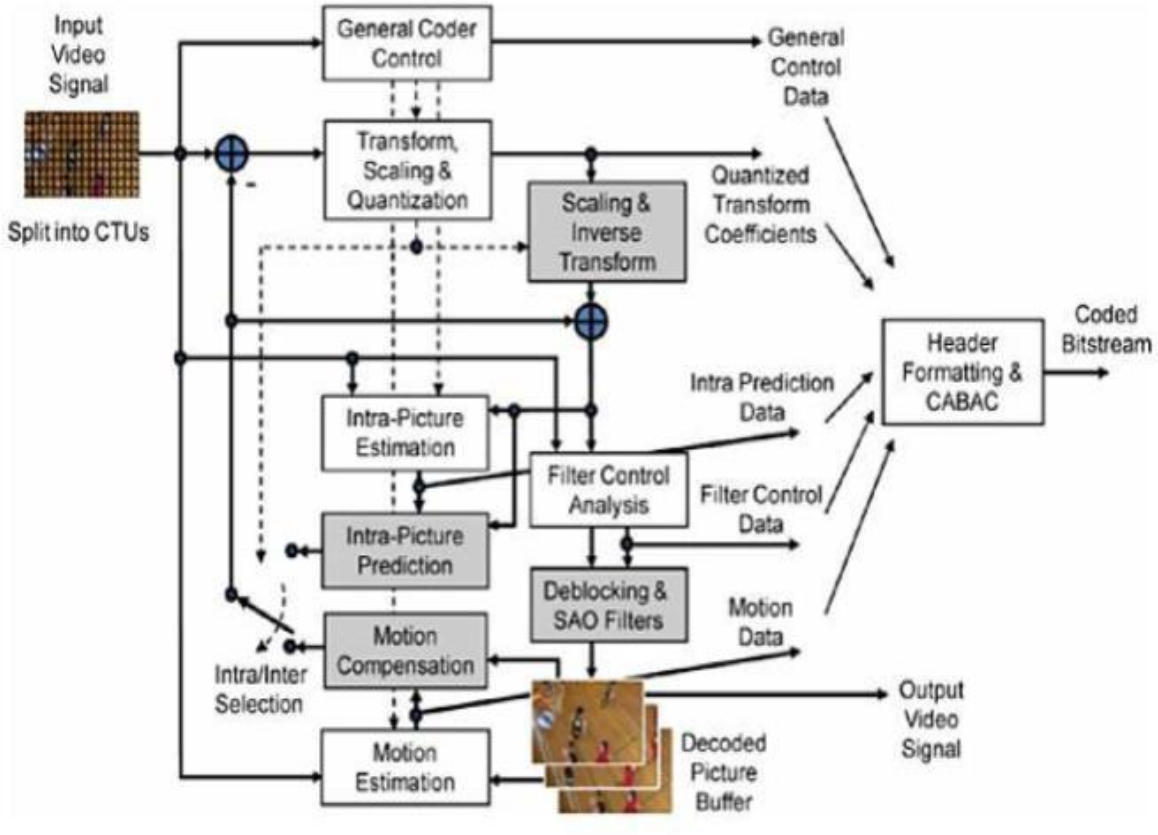


Figura 2.8 Diagrama en bloques del codificador HEVC/H.265 [48].

Debido a la alta complejidad de los algoritmos introducidos por H.265, el estándar integra novedosas herramientas con el objetivo de viabilizar la codificación de la imagen. HEVC permite la codificación del Frame en dos nuevos tipos adicionales al *slice*, que existía previamente en los anteriores estándares, denominados *Tile* y Frente de Ondas o WPP (*Wavefront Parallel Processing*).

- *Tile*: define regiones descifrables rectangulares de la imagen, típicamente con el mismo número de CTU, éstas pueden ser codificadas independientemente y comparten algunas cabeceras de información. Los *tiles* tienen como propósito fundamental aumentar la capacidad para el procesamiento paralelo en lugar de proporcionar la capacidad de recuperación de errores. MúltIples *tiles* pueden compartir información de cabecera por estar contenida en el mismo *slice*. Alternativamente, un solo *tile* puede contener múltIples *slices*. Los *tiles*

proporcionan cierto nivel de paralelización en los procesos de codificación y decodificación de video que puede ser explotado en sistemas con arquitecturas de procesamiento paralelo, siendo más flexibles que las *slices* de H.264. Además, se considera una tecnología menos compleja que FMO (*Flexible Macroblock Ordering*).

- WPP: Es una herramienta utilizada por HEVC en el procesamiento en paralelo. Cuando WPP (*Wavefront Parallel Processing*) se encuentra habilitado, una *slice* se fracciona en filas de CTU. La primera fila puede ser procesada por un hilo de ejecución; en el caso de un segundo hilo de ejecución puede comenzar con la segunda fila después de haberse procesado las dos primeras CTU de la fila anterior. Al igual que los *tiles*, las WPP suministran una forma de paralelización en los procesos de codificación y decodificación. La diferencia radica en que con WPP se consigue un mejor rendimiento de compresión además de no permitir la introducción de determinadas aberraciones visuales como ocurre en los *tiles* [48].

El codificador x265 es disponible como una biblioteca de código abierto, publicada bajo la licencia GPLv2. También está disponible bajo una licencia comercial, permitiendo a las compañías comerciales utilizar y distribuir x265 en sus soluciones sin estar sujeto a las restricciones de la licencia GPL. x265 es desarrollado por *MulticoreWare*, líderes en soluciones de *software* de alto rendimiento, con respaldo de video, líder en proveedores de tecnología, incluidos *Telestream* y *Doremi Labs*. Si bien x265 está diseñado principalmente como una biblioteca de *software* de codificador de video, se proporciona un ejecutable de línea de comando para facilitar las pruebas y el desarrollo [49], [50].

x265 tiene una serie de opciones predefinidas que hacen intercambios entre la velocidad de codificación (cuadros codificados por segundo) y la eficiencia de compresión (calidad por bit en el flujo de bits). El preajuste predeterminado es *medium*. Provee un equilibrio razonable entre calidad posible y el gasto de ciclos de CPU. A medida que los perfiles se desplazan hacia la derecha (ver figura 2.9) se intensifica el uso de funciones y herramientas que posibilitan mayores niveles de compresión y calidad de la imagen codificada, a expensas del gasto computacional. A medida que se avanza hacia la izquierda se maximiza

la rapidez en el proceso de codificación a expensas de la calidad de la imagen y el *bitrate* de la imagen codificada.

	ultra-fast	super-fast	very-fast	faster	fast	medium	slow	slower	verys-low	placebo
ctu	32	32	32	64	64	64	64	64	64	64
min-cu-size	16	8	8	8	8	8	8	8	8	8
bframes	3	3	4	4	4	4	4	8	8	8
b-adapt	0	0	0	0	0	2	2	2	2	2
rc-lookahead	5	10	15	15	15	20	25	30	40	60
scenecut	0	40	40	40	40	40	40	40	40	40
refs	1	1	1	1	2	3	3	3	5	5
me	dia	hex	hex	hex	hex	hex	star	star	star	star
merange	57	57	57	57	57	57	57	57	57	92
subme	0	1	1	2	2	2	3	3	4	5
rect	0	0	0	0	0	0	1	1	1	1
amp	0	0	0	0	0	0	0	1	1	1
max-merge	2	2	2	2	2	2	3	3	4	5
early-skip	1	1	1	1	0	0	0	0	0	0
fast-intra	1	1	1	1	1	0	0	0	0	0
b-intra	0	0	0	0	0	0	0	1	1	1
sao	0	0	1	1	1	1	1	1	1	1
signhide	0	1	1	1	1	1	1	1	1	1
weightp	0	0	1	1	1	1	1	1	1	1
weightb	0	0	0	0	0	0	0	1	1	1
aq-mode	0	0	1	1	1	1	1	1	1	1
cuTree	0	0	0	0	1	1	1	1	1	1
rdLevel	2	2	2	2	2	3	4	6	6	6
rdoq-level	0	0	0	0	0	0	2	2	2	2
tu-intra	1	1	1	1	1	1	1	2	3	4
tu-inter	1	1	1	1	1	1	1	2	3	4

Figura 2.9. Perfiles de codificación de H.265 [51].

En HEVC usan un CRF con un rango de medición de calidad entre 0 a 51, siendo 0 lo mejor y 51 lo peor. Por defecto usa 28, el cual no tiene nada que ver con la medición de H.264. Pero se recomienda 23 (que vendría a ser el equivalente de 18 en H.264). Los preset de compresión de calidad son los mismos, por defecto usa medium, pero ya que se quiere realizar una transmicion en tiempo real se usará ultrafast [4].

En [48] se expone el porqué de escoger ultrafast, ya que en el caso de los tiempos de codificación se puede observar que los menores tiempos se obtienen para el perfil *ultrafast.* Primeramente, se debe aclarar que si el hecho de poseer CTU de 64x64 es una ventaja

cuando se necesita un nivel de compresión alto, esto es una limitación a la hora de realizar el procesamiento paralelo de la secuencia de imágenes con el WPP. En el caso de *medium* y *placebo* utilizan la mitad de las filas de procesamiento paralelo que *ultrafast*, en todas las muestras (por ejemplo, en la muestra de resolución estándar *ultrafast* utiliza 68 filas mientras que *medium* y *placebo* 34). En *ultrafast* se encuentra habilitada la opción *fast-intra*, lo que implica que solo son chequeados entre 8 y 10 vectores en saltos de 5 o 6. La predicción intra-angular realizada se basa en la predicción de los vectores de mejor desempeño. En los perfiles *medium* y *placebo* se analizan los 33 vectores de predicción direccional. Además, en *ultrafast* no se realiza la colocación forzada de imágenes I (*scenecut*=0) por lo que el búffer de colocación anticIpada (*Lookahead*) tiene un tamaño pequeño, incidiendo de forma determinante en la latencia del proceso de codificación. *Ultrafast* posee una estructura fija de GOP (*Group of Pictures*) con 3 imágenes tipo B en cada GOP. Ello implica que el nivel de esfuerzo del codificador para determinar la colocación de este tipo de imágenes es mínimo. En el caso de *medium* y *placebo* se utilizan algoritmos de Trellis para determinar la colocación de imágenes tipo B, lo que incide directamente en el monto total del proceso de codificación [52].

Con HEVC se obtiene menos peso, pero también puede relentizar la reproducción dependiendo del equipo usado. Si se hizo un encodeo lento con configuraciones que priorizan la calidad, eso puede afectar la decodificación [53].

2.7 Para reproducir en la web

Al servicio *streaming* con codificación de video H.265 se puede acceder sin ningún problema desde los reproductores de video solo con poner la URL del servidor de video siempre y cuando estos sean capaces de reproducir este codec; pero este servicio *streaming* se quiere implementar para una plataforma web por lo que es indispensable el uso de los navegadores web, sin embargo estos aún no tienen disponible la reproducción de videos con codificación H.265 por lo que tambien se hará disponible la tranmisión en H.264.

2.8 Conclusiones

A lo largo de este capítulo se explicaron cuales son las herramientas con las que cuenta la plataforma de *streaming*, cómo acceder a ella como cliente (para ver el servicio *streaming* desde un navegador o reproductor) y como administrador . Se hizo un análisis del formato de compresión de video H.265. Además se esbozaron algunos conceptos relativos a la transmisión de *streaming* sobre la red, destacando las etapas involucradas en el proceso.

CAPÍTULO 3. Resultados y discusión

3.1 Introducción

En el presente capítulo se realizará el análisis y la discusión de los resultados obtenidos al implementar el servicio *streaming* en la red universitaria durante una etapa de prueba para así observar el alcance y las métricas del *streaming*. Para esto se describirá la estructura del servicio *streaming*, se mostrarán algunas de las variantes de configuración del mismo y se analizarán los resultados a partir de la herramienta Ganglia. Al final del capítulo se mencionan algunos de los problemas que se pueden presentar a la hora de crear el servicio *streaming*.

3.2 Estructura del servicio *streaming*

El servicio *streaming* final tendrá una arquitectura típica (servidor-cliente) con descarga progresiva y será *streaming* directo.

Las propiedades físicas del servidor como se muestra en la figura 3.1 son 8 núcleos que trabajan a 3.16 GHz con una memoria RAM de 11.70 GB con Ubuntu 16.04 como sistema operativo

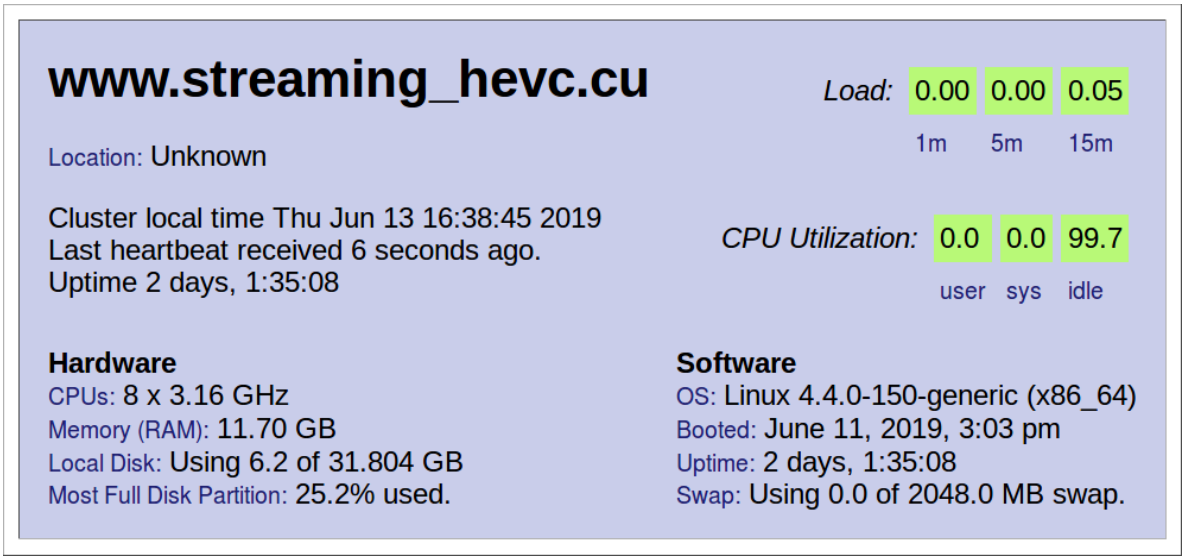


Figura 3.1 Características del servidor

3.2.1 Captura de video

La grabación se podrá hacer desde un móvil usando aplicaciones realizadas con ese propósito, desde cámaras Ip o desde la misma webcam de un pc. Los principales protocolos usados serán UDP, TCP y RTMP. Es importante señalar que para al envió del video se puede contar el ***VLC***, obs o con el ***FFmpeg***.

Durante el tiempo de prueba la captura de video se realiza desde la webcam de un pc con la ayuda de ***FFmpeg*** a través del protocolo UDP ya que UDP no es orientado a la conexión y si TCP, haciendo más lento el proceso de conexión al servidor.

3.2.2 Servidor

Los servidores de video usados fueron ***FFserver*** el cual es una herramienta de ***FFmpeg*** y ***Nginx*** que puede incluir en su configuración la codificación de video auxiliándose de ***FFmpeg***.

Para el tiempo de prueba se tomó ***FFserver*** como servidor ya que permite la codificación de video con el códec H.264 y H.265. También se realizó la implementación del servicio sobre la plataforma ***Nginx***.

Nginx es un servidor web/proxy inverso ligero de alto rendimiento y un proxy para protocolos de correo electrónico (IMAP/POP3). Es software libre y de código abierto,

licenciado bajo la Licencia BSD simplificada; también existe una versión comercial distribuida bajo el nombre de ***Nginx*** plus. Es multiplataforma, por lo que corre en sistemas tipo Unix y Windows. Es usado por una larga lista de sitios web, como: WordPress, Netflix, Hulu, GitHub, Ohloh, SourceForge.[54].

Entre sus usos está el *streaming* de archivos flv y mp4 para lo cual cuenta con módulos como su módulo RTMP.

En los anexos II, III y IV se puede ver las configuraciones de ***FFserver*** y de ***Nginx***. Donde ***Nginx*** trabaja con RTMP y ***FFserver*** con http.

Para crear la página web con la que se presentará a los usuarios la plataforma *streaming* se usa el servidor *apache*.

Apache es el servidor usado para realizar la interfaz del sitio web del servidor, este un software de servidor web gratuito y de código abierto para sistemas operativos modernos, incluyendo UNIX y Windows, con el cual se ejecutan el 46% de los sitios web de todo el mundo. El nombre oficial es *Apache HTTP Server*, y es mantenido y desarrollado por la *Apache Software Foundation*. El objetivo de este proyecto es proporcionar un servidor seguro, eficiente y extensible que brinde servicios HTTP en sincronización con los estándares HTTP actuales [55].

El cliente podrá acceder al servicio *streaming* desde la dirección Ip del servidor o desde su url tanto desde un navegador como desde un reproductor de video

3.2.3 Codificación de video

El video para el *streaming* se codifica con la herramienta ***FFmpeg*** que tiene las librerías de códec necesarias para comprimir el video recibido usando H.264 y H.265 como códec de video, aac como códec de audio y contenedores como asf, mp4 y flv.

A pesar de ser ***FFmpeg*** quien codifica es importante señalar que tanto en ***FFmpeg*** como en ***Nginx*** y en ***FFserver*** se deben definir aspectos como la resolución, razón de bits, velocidad de codificación, buffers de audio y video, metadatos, filtros, códec, contenedores, etc. Donde dichos aspectos deben coincidir.

3.3 Análisis del *streaming*

Las pruebas se realizarán para codificación con H.265 y H.264 y con ambas juntas. El video tendrá las siguientes características

Tabla 3.1 Principales parámetros del video codificado

Códec de audio	MPEG AAC Audio (mp4a)	MPEG AAC Audio (mp4a)
Canales de audio	Estéreo	Estéreo
Tasa de muestra	44100 Hz	44100 Hz
Códec de Video	MPEG-H Part2/ HEVC (H.265)	H.264 MPEG-4 AVC (part 10)
Resolución de pantalla	1280x720 (hd720)	1280x720 (hd720)
Tasa de Fotogramas	30	30
Formato de codificación	Planar 4:2:2 YUV	Planar 4:2:2 YUV

Luego de realizar el *streaming* con codificación H.264 y H.265 obtenemos la siguiente imagen (figura 3.2) donde se puede apreciar la mejora en la calidad de la imagen con codificación H.265.

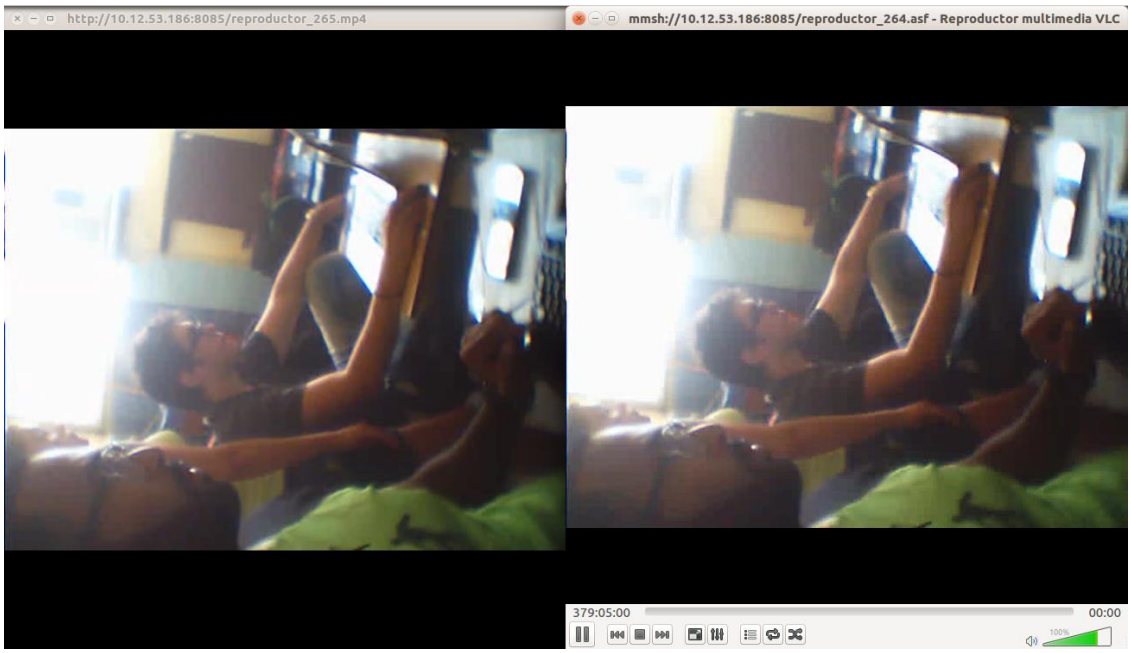


Figura 3.2 Captura de imagen con codificación H.265 y H.264

En la figura 3.2 y en la tabla 3.2 se ve como al analizar dos muestras de video de 1 minuto cada una con codificación H.264 y H.265 se logra un video con menor peso al codificar con H.265.

Tabla 3.2 Muestras de video durante un minuto.

	Códec	H.264	H.265
Audio	Decodificados (bloques)	2770	3003
	Reproducidos (buffers)	2770	3003
	Perdidos (buffers)	0	0
Video	Decodificados (bloques)	1924	1895

	Mostrados (fotogramas)	1898	1891
	Perdidos (fotogramas)	0	0
Entrada/Lectura	Tamaño de datos del medio (KiB)	7113	4759
	Tamaño de bits de entrada (kb/s)	726	583
	Tamaño de datos demuxados (KiB)	6911	4128
	Tasa de bits (kb/s)	703	1148
	Descartados	0	0

3.4 Ganglia

Ganglia es una herramienta de software de monitoreo distribuido escalable de código abierto para sistemas de computación de alto rendimiento que se puede usar para ver métricas de estadísticas en vivo, como la carga promedio de CPU y la utilización de la red para múltiples sistemas.

Ganglia utiliza XML para la representación de datos, RRDtool para el almacenamiento y visualización de datos y XDR para el transporte de datos compacto y portátil. Utiliza estructuras de datos cuidadosamente diseñadas para lograr gastos generales muy bajos y alta concurrencia [56].

Ganglia consta de tres componentes principales: *Ganglia Monitoring Daemon*, *Ganglia Meta Daemon* y *Ganglia PHP Web Front-end*.

1. *Ganglia Monitoring Daemon*: escucha el canal de mensajes del clúster y almacena los datos en la memoria y responde a las solicitudes de una descripción XML del estado del clúster. Gmond se ejecuta en cada sistema que desea monitorear y monitorear los cambios en el estado del host.
2. *Ganglia Meta Daemon*: *Ganglia Meta Daemon* se ejecuta en los nodos maestros y recopila información de múltiples máquinas cliente.
3. *Ganglia PHP Web Front-end*: se utiliza para mostrar toda la información recopilada de las máquinas cliente a través de páginas web.

Durante un período de prueba de aproximadamente 2 horas donde los períodos de análisis fueron:

Entre las 1:33 pm y las 2:05 pm con codificación H.264.

Entre las 2:15 pm y las 2:40 pm con codificación H.265.

Entre las 2:50 pm y las 3:10 pm con ambas.

Al concluir el período de prueba y obtener las métricas de desempeño disponibles en el Ganglia disponible en http://10.12.53.186/ganglia se obtienen los principales parámetros relativos al servidor mostrados en las figuras 3.3 a la 3.11 y en los anexos del V al IX.

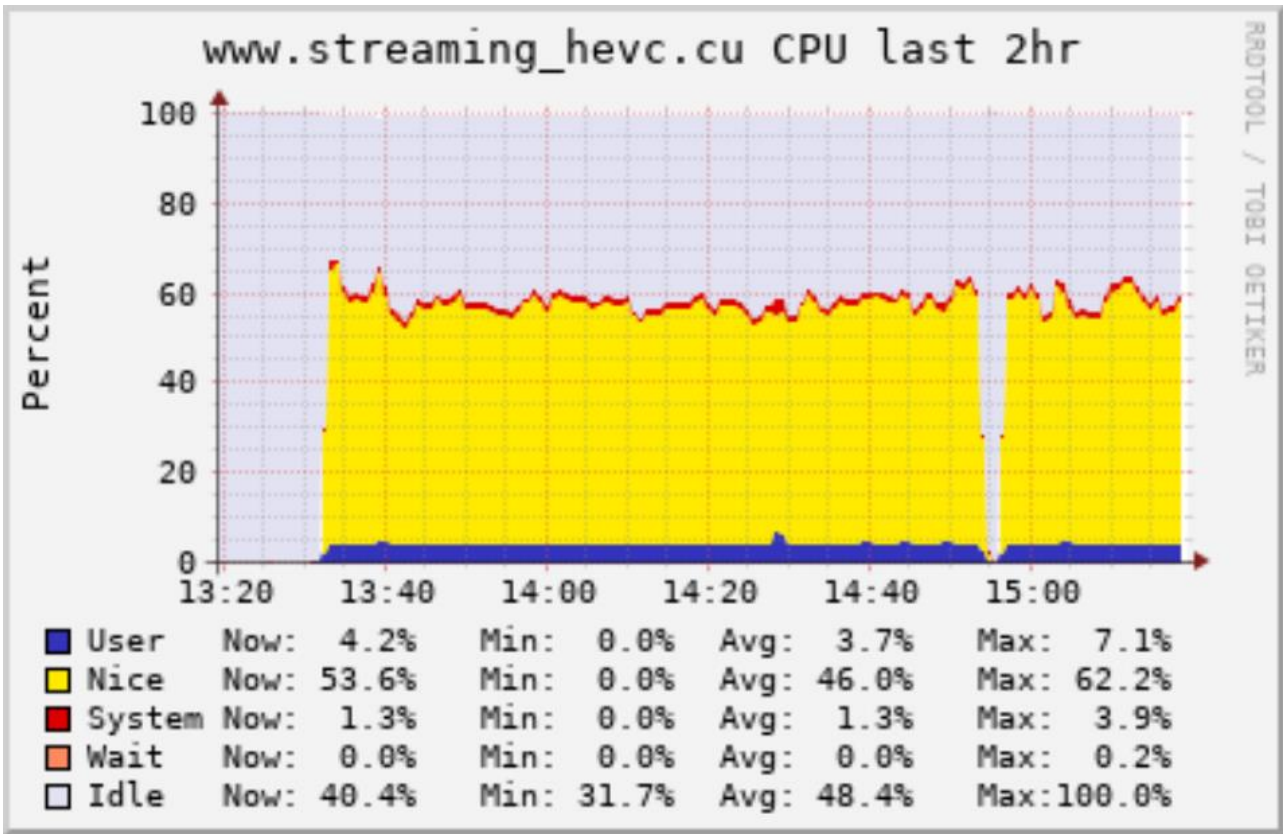


Figura 3.3 Carga del CPU.

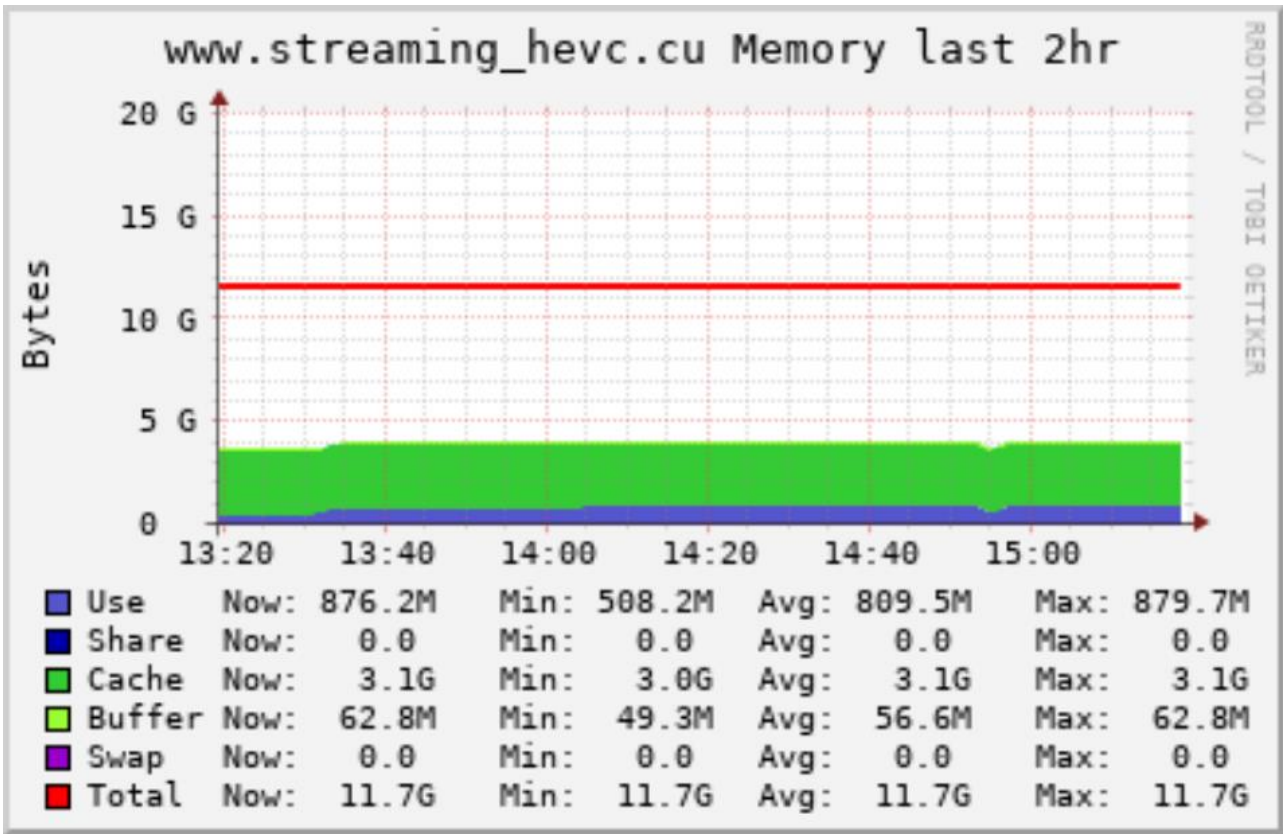


Figura 3.4 Memoria utilizada.

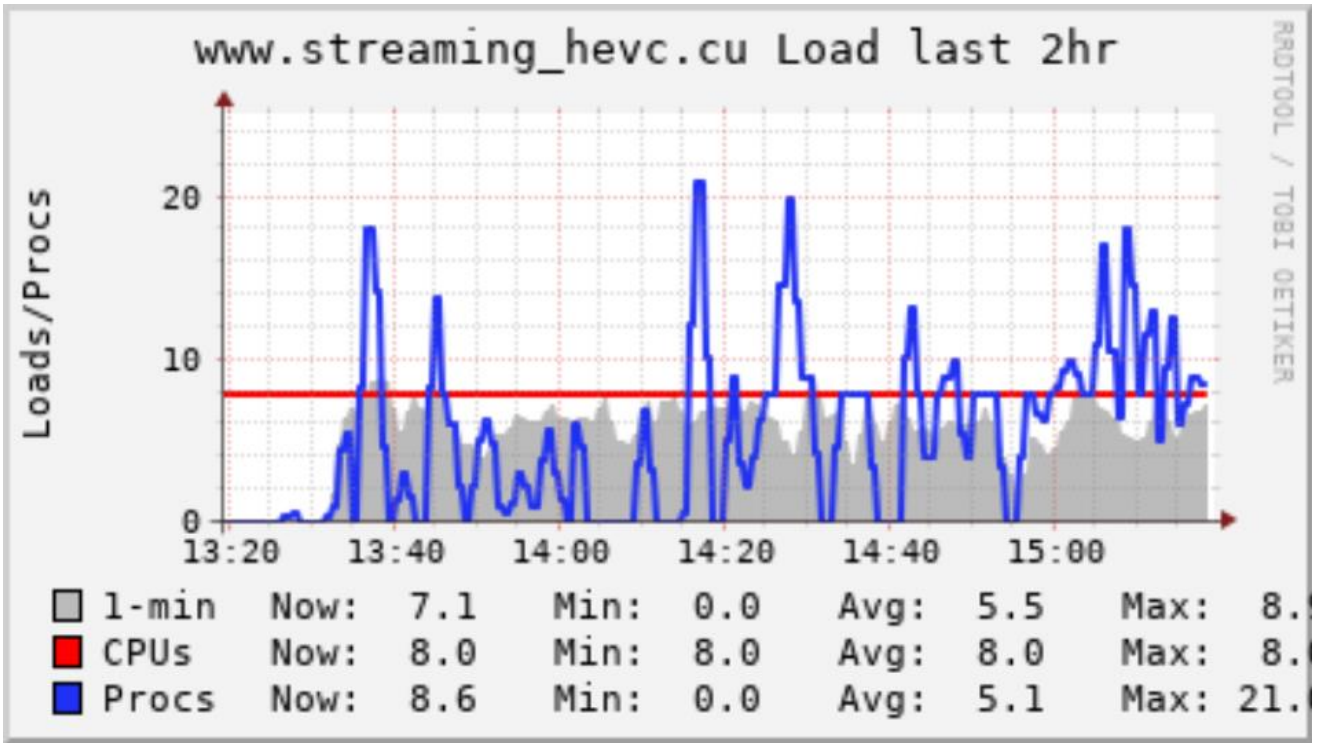


Figura 3.5 Carga de los procesos y del CPUs.

En la figura 3.3 y 3.5 se observa como la carga de la CPU permanece prácticamente constante durante el periodo de prueba, pero la carga de los procesos durante la codificación con H.265 es superior a H.264 ya que para codificar con H.265 se usan algoritmos más complejos y el procesador trabaja más.

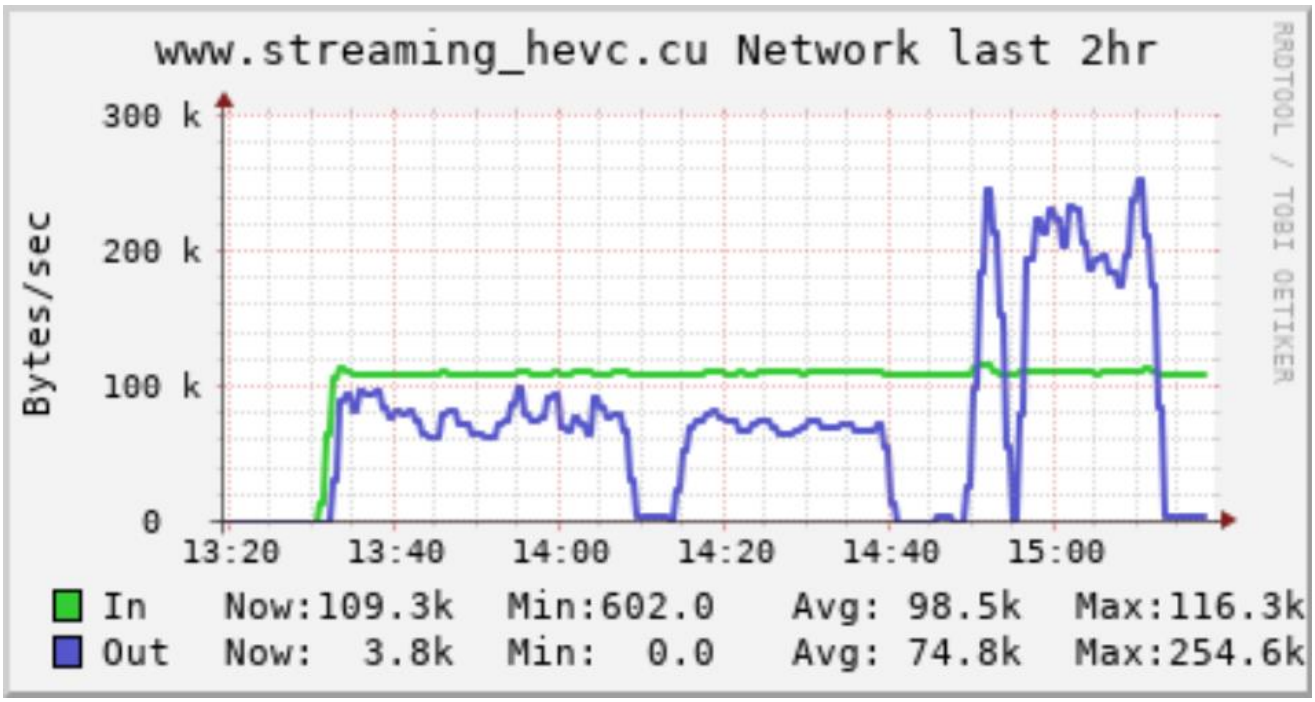


Figura 3.6 Tráfico de red.

En la figura 3.6 se observa como el tráfico de la red de entrada permanece constante ya que el flujo de entrada es constante mientras que el tráfico de salida no porque el codificador codifica con bitrate variable. También se observa que al codificar con H.265 es menor el trabajo en el tráfico de la red a la salida.

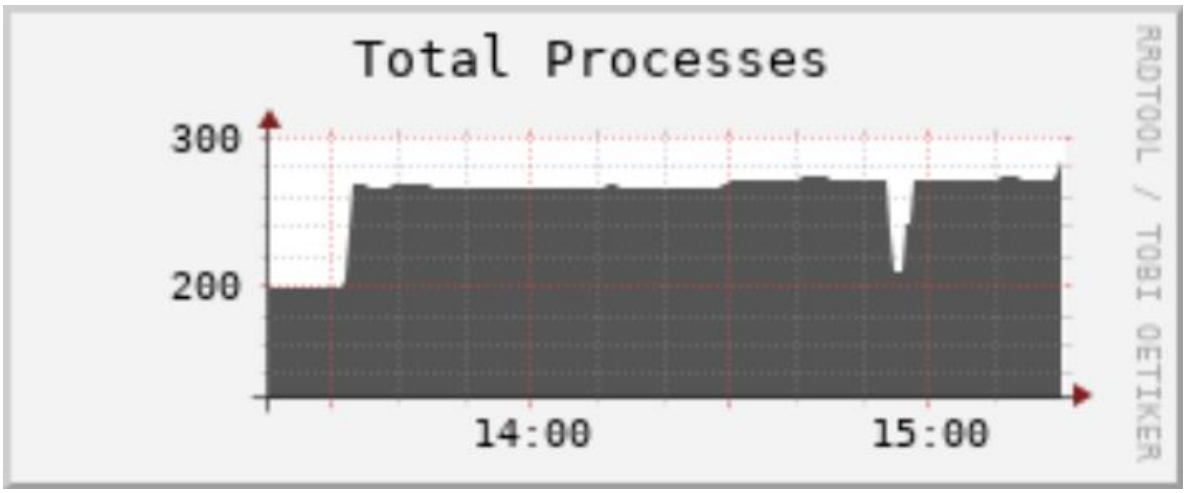


Figura 3.7 Procesos totales.

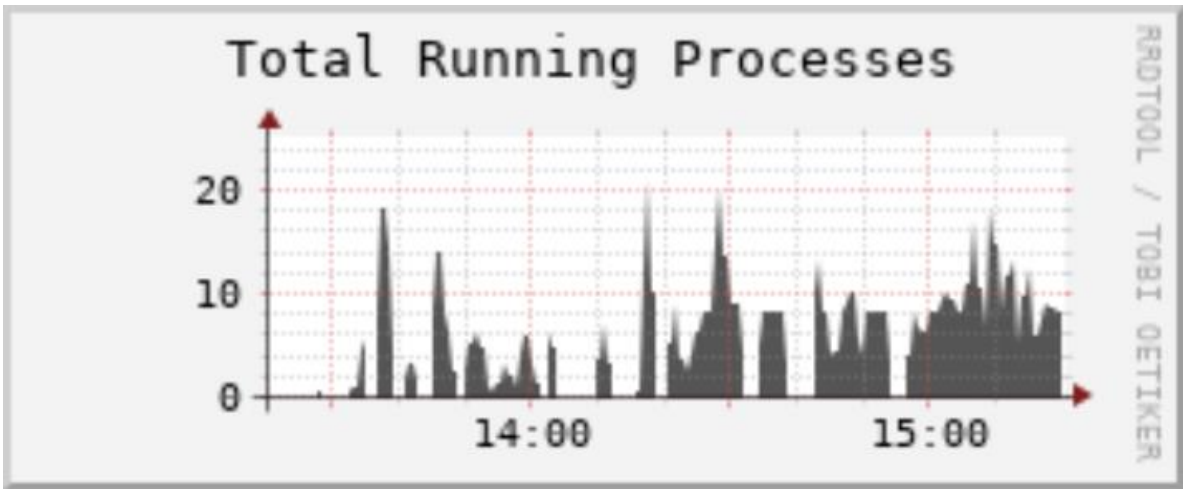


Figura 3.8 Procesos corriendo.

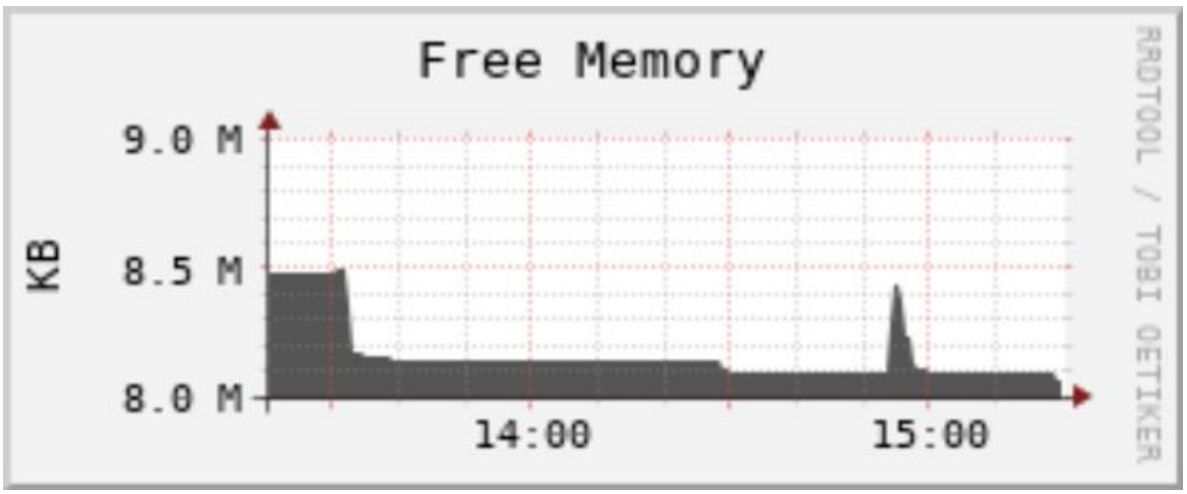


Figura 3.9 Memoria libre.

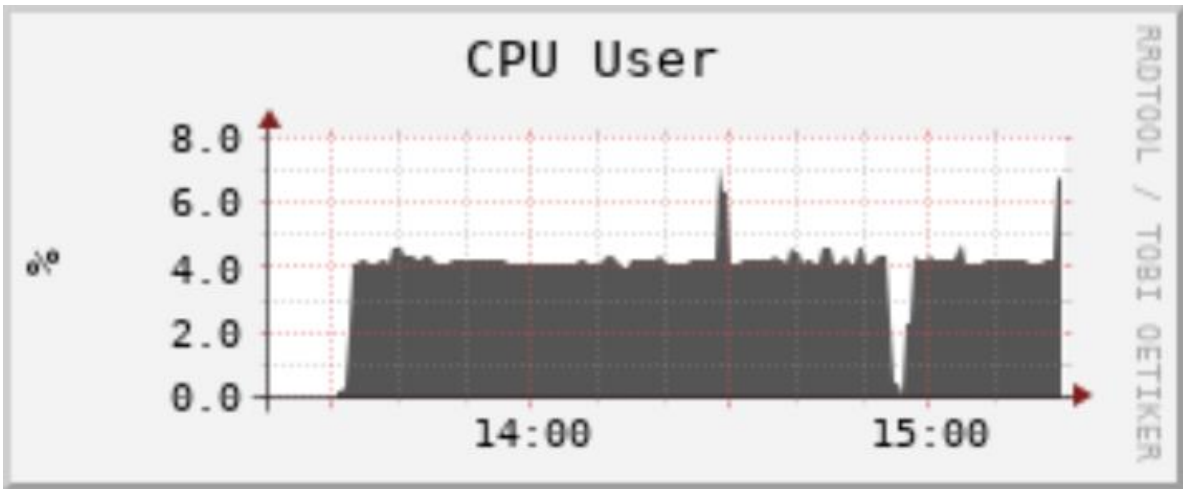


Figura 3.10 Porciento de utilización de la CPU.

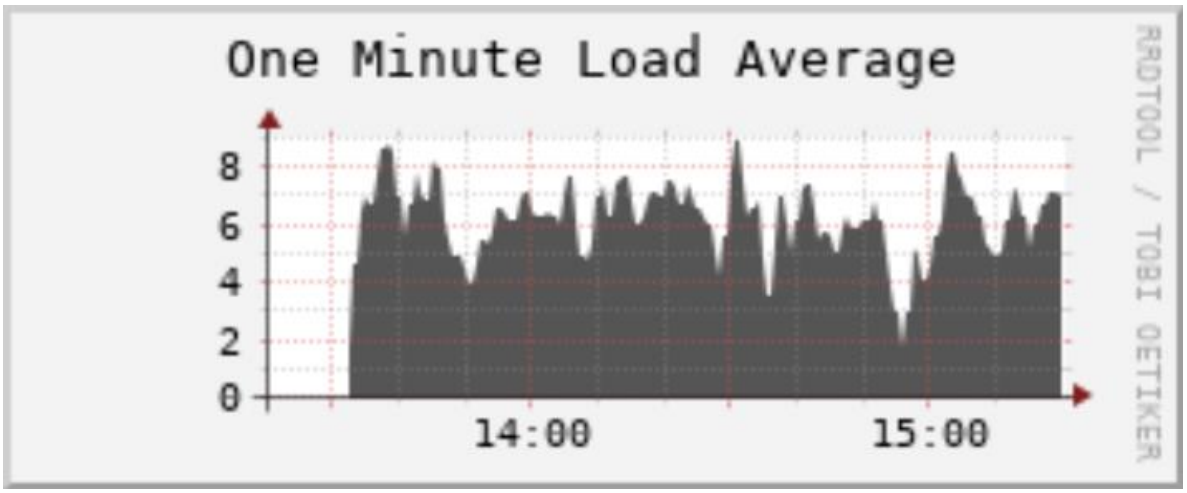


Figura 3.11 Carga promedio con resolución de un minuto

3.5 Problemas durante el diseño

Entre los principales problemas encontrados durante la realización del trabajo están:

- El problema con el audio ya que afectó mucho el ruido ambiente y el eco de la misma grabación. Para eliminarlo se aplicó un filtro pasabanda entre los 200 Hz y 800 Hz, y se modificó la captura de audio con el uso del comando **alsamixer**.
- El servidor ***Nginx*** a pesar de ser más practico que ***FFserver*** no codifica H.265 principal objetivo de este trabajo por lo que el streaming de video se realizó co ***FFserver*** y no con ***Nginx*** a pesar de haberse realizado el streaming con ***Nginx***, pero solo con codificación H.264.

- ***FFserver*** no está presente en las últimas versiones de ***FFmpeg.***
- Problemas con los repositorios donde se señala que para el trabajo de Ubuntu es necesario una conexión directa a internet.

Es importante señalar que la principal ayuda para el entendimiento y realización del servicio *streaming* fue encontrada en los blogs de páginas webs.

3.6 Conclusiones del capítulo

Al finalizar este capítulo se logró realizar el servicio *streaming*, donde se analizó el mismo durante varios períodos de prueba alcanzándose los resultados esperados.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

Luego de realizar la investigación el autor llega a las siguientes conclusiones:

1. El *streaming* es una tecnología de transmisión a través de la red, en la que no existe descarga de la información en un disco local, donde la información enviada al cliente es reproducida en tiempo real una vez que es recibida. Donde el *streaming* según el tipo de descarga se puede clasificar en descarga progresivo o por segmento y puede ser *streaming* directo o bajo demanda. Para la realizar el servicio *streaming* es importante conocer el ancho de banda disponible, los formatos usados para la codificación y tener conocimientos sobre el servidor que se va a utilizar.
2. Para realizar un *streaming* de video en tiempo real es necesario contar con equipos capaces de capturar y codificar el video en tiempo real. La captura de video se realiza desde la webcam de la computadora, desde una cámara Ip o desde un móvil que funcione como una cámara Ip. El video es enviado al servidor por ***FFmpeg***, donde el servidor de video usado es ***FFserver*** y ***Nginx***. El video es codificado por ***FFmpeg*** en H.265 y H.264. Los usuarios podrán acceder al servicio *streaming* desde un reproductor de video o desde la página web del *streaming* que se realiza usando el servidor *apache*.
3. El servidor *streaming* se realizó usando ***FFmpeg***, ***FFserver*** y el servidor *apache* donde luego de un período de prueba se midieron las métricas de desempeño usando Ganglia y los resultados obtenidos probaron la superioridad de H.265 sobre H.264 así como el buen funcionamiento del servicio streaming en ambos códec.

Recomendaciones

El autor recomienda los siguientes aspectos:

1 Buscar nuevas alternativas para mejorar el servicio *streaming* propuesto en este trabajo.

2 Que con la realización de este trabajo se permita en el futuro la creación de un canal de video en la uclv con carácter interdisciplinario.

REFERENCIAS BIBLIOGRÁFICAS

[1] Á. Romero Alaguna y S. Ramírez Velázquez, «Diseño de una arquitectura de streaming para redes mesh para entornos de bajos recursos en Colombia», Universidad Libre, Facultad de Ingeniería; Ingeniería de Sistemas, Bogotá D.C, 2016.

[2] Juliana Garzón Vásquez, «"Streaming de video por telefonía móvil"», Universidad Distral Francisco José de Caldas Facultad de Ingeniería; Ingeniería electronica, Bogotá, D.C., 2017.

[3] L. Calle y V. del Rocío, «Implementación de un servidor de video streaming basado en los sistemas de gestión de aprendizaje LMS.», Tesis de grado, Universidad Nacional de Loja Facultad de Energía, las Industrias y los Recursos Naturales no Renovables; Ingeniería en Telecomunicaciones, 2017.

[4] Y. López Pérez, «Análisis de la calidad perceptual de video utilizando el estándar de compresión H.265/HEVC», Thesis, Universidad Central «Marta Abreu» de Las Villas, Facultad de Ingeniería Eléctrica, Departamento de Electrónica y Telecomunicaciones, 2017.

[5] M. Martínez y P. Emilio, «Compresión eficiente de materiales audiovisuales utilizando H.265-HEVC», Thesis, Universidad Central «Marta Abreu» de Las Villas . Facultad de Ingeniería Eléctrica. Departamento de Electrónica y Telecomunicaciones, 2016.

[6] P. Cuéllar y S. Wilfredo, «Codificación de video de alta eficiencia en un clúster de computadoras para implementarse en la televisión digital de ultra alta definición», Thesis, Universidad Central «Marta Abreu» de Las Villas . Facultad de Ingeniería Eléctrica. Departamento de Electrónica y Telecomunicaciones, 2016.

[7] «Video streaming», 2012.

[8] «¿Qué es el streaming?», 2008. [En línea]. Disponible en: http://www.ite.educacion.es/formacion/materiales/107/cd/video/video0103.html. [Accedido: 09-abr-2019].

[9] Mariano G, «La realidad actual del streaming de video. El streaming tradicional vs alternativas actuales», Universidad Austral, 2013.

[10] Francisco José Suárez Alonso, «Tecnologías de Streaming», Oviedo, 2010.

[11] C. Xu, C. Xiong, y J. J. Corso, «Streaming Hierarchical Video Segmentation», en *Computer Vision – ECCV 2012*, 2012, pp. 626-639.

[12] Mike Colburn, «Diferencia entre la descarga progresiva y streaming de video.», 06-jun-2016. [En línea]. Disponible en: http://abcarticulos.info/article/diferencia-entre-la-descarga-progresiva-y-streaming-de-video-en-lnea. [Accedido: 09-abr-2019].

[13] «¿Qué es un CDN? ¿Y un servidor en la nube?», *HostingExperto*, 03-may-2016. .

[14] M. Denies, «Streaming panoramic video», WO2001095513A1, 13-dic-2001.

[15] DAVID AUSTERBERRY, *LA TECNOLOGIA DEL STREAMING DE VIDEO Y AUDIO*. ESCUELA DE CINE Y VIDEO DE ANDOAIN, 2005.

[16] X Basogain, «Cursos en vídeo de alta calidad», *ResearchGate*, 2004. [En línea]. Disponible en: https://www.researchgate.net/publication/28067029_Cursos_en_video_de_alta_calidad. [Accedido: 09-abr-2019].

[17] Y. T. Hou, «Streaming video over the Internet: approaches and directions», *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, n.º 3, pp. 282-300, mar. 2001.

[18] R W Lowell, «Internet event timer recording for video and/or audio», 2000.

[19] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, y Y. A. Reznik, «Video coding for streaming media delivery on the Internet», *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, n.º 3, pp. 269-281, mar. 2001.

[20] S.-J. Park, «Method for transmitting moving picture data to mobile terminal using pseudo-streaming technology», US20060200577A1, 07-sep-2006.

[21] Y. Liu, F. Li, L. Guo, B. Shen, y S. Chen, «A Comparative Study of Android and iOS for Accessing Internet Streaming Services», en *Passive and Active Measurement*, 2013, pp. 104-114.

[22] T. Stockhammer, «Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles», en *Proceedings of the Second Annual ACM Conference on Multimedia Systems*, New York, NY, USA, 2011, pp. 133–144.

[23] S. Lederer, C. Müller, y C. Timmerer, «Dynamic Adaptive Streaming over HTTP Dataset», en *Proceedings of the 3rd Multimedia Systems Conference*, New York, NY, USA, 2012, pp. 89–94.

[24] Mariano García Clavería, «La realidad actual del streaming de video. El streaming tradicional vs alternativas actuales», Universidad Austral, 2013.

[25] Brijesh Kumar, «Making Sense of Video streaming Formats», 2016.

[26] «ANÁLISIS DE TECNOLOGÍAS DE STREAMING: EVALUACIÓN DE PROTOCOLOS Y DISEÑO DE UN CASO DE ESTUDIO - PDF». [En línea]. Disponible en: https://docplayer.es/52592352-Analisis-de-tecnologias-de-streaming-evaluacion-de-protocolos-y-diseno-de-un-caso-de-estudio.html. [Accedido: 04-jun-2019].

[27] marilin, «Códecs de vídeo: ventajas y desventajas de los diferentes tipos», *Hipertextual*, 16-jul-2012. [En línea]. Disponible en: https://hipertextual.com/archivo/2012/07/codecs-ventajas-desventajas-diferentes-tipos/. [Accedido: 08-may-2019].

[28] marilin, «Guía básica sobre contenedores, códecs y formatos de vídeo», *Hipertextual*, 09-jul-2012. [En línea]. Disponible en: https://hipertextual.com/archivo/2012/07/guia-basica-sobre-contenedores-codecs-y-formatos-de-video/. [Accedido: 08-may-2019].

[29] R. Antolín Prieto y R. Antolín Prieto, «YouTube como paradigma del vídeo y la televisión en la web 2.0», info:eu-repo/semantics/doctoralThesis, Universidad Complutense de Madrid, Madrid, 2012.

[30] James Bennett, «The Netflix Prize», *ResearchGate*, 01-ene-2009. [En línea]. Disponible en: https://www.researchgate.net/publication/223460275_The_Netflix_Prize. [Accedido: 09-abr-2019].

[31] «VLC: Sitio oficial - ¡Soluciones multimedia libres para todos los sistemas operativos! - VideoLAN». [En línea]. Disponible en: https://www.videolan.org/index.es.html. [Accedido: 02-may-2019].

[32] M. Camuñas, «📽 22 plataformas y herramientas para emitir en streaming», *Max Camuñas*, 18-dic-2018. [En línea]. Disponible en: https://www.maxcf.es/emitir-en-streaming/. [Accedido: 10-abr-2019].

[33] Mauricio Venegas Morales, «Capturar Video en GNU LINUX RCA», *Scribd*. [En línea]. Disponible en: https://es.scribd.com/document/341890339/Capturar-Video-en-GNU-LINUX-RCA. [Accedido: 10-abr-2019].

[34] «QUE ES Y COMO FUNCIONA UNA CAMARA IP - TVCenLinea.com». [En línea]. Disponible en: http://www.foro.tvc.mx/kb/a551/que-es-y-como-funciona-una-camara-ip.aspx. [Accedido: 02-may-2019].

[35] J. Villegas, «¿Cómo configurar una Cámara IP en una red LAN?» [En línea]. Disponible en: https://www.tecnoseguro.com/tutoriales/cctv/como-configurar-una-camara-ip-en-una-red-lan. [Accedido: 02-may-2019].

[36] «DroidCam | Dev47Apps». .

[37] «Connect | Dev47Apps». .

[38] «IP Webcam, convierte tu Android en una Web Cam - Sobre Android». [En línea]. Disponible en: http://sobreandroid.com/ip-webcam-convierte-tu-android-en-una-web-cam/. [Accedido: 08-may-2019].

[39] atareao, «Utiliza tu Android como webcam en Ubuntu», *El atareao*. .

[40] «IP Webcam - Apps en Google Play». [En línea]. Disponible en: https://play.google.com/store/apps/details?id=com.pas.webcam&hl=es_419. [Accedido: 08-may-2019].

[41] J. G. Nieto, «Cómo usar la cámara de tu móvil Android como webcam para tu PC», *Xataka Android*, 15-oct-2018. [En línea]. Disponible en: https://www.xatakandroid.com/tutoriales/como-usar-camara-tu-movil-android-como-webcam-para-tu-pc. [Accedido: 02-may-2019].

[42] R. Andrés, «Cómo convertir tu móvil Android en una Webcam», *ComputerHoy*, 12-ago-2017. [En línea]. Disponible en: https://computerhoy.com/paso-a-paso/moviles/como-convertir-tu-movil-android-webcam-66067. [Accedido: 02-may-2019].

[43] «FFmpeg: codificación audio/vídeo por línea comandos [uso avanzado]», *Gnu/Linux Vagos*. [En línea]. Disponible en: https://gnulinuxvagos.es/topic/5789-ffmpeg-codificaci%C3%B3n-audiov%C3%ADdeo-por-l%C3%ADnea-comandos-uso-avanzado/. [Accedido: 02-may-2019].

[44] R. A. Olivera Solís y V. Alfonso Reguera, *Plataforma para codificación en HEVC/H.265*. 2017.

[45] «ffserver – FFmpeg». [En línea]. Disponible en: https://trac.ffmpeg.org/wiki/ffserver. [Accedido: 02-may-2019].

[46] F. Bossen, B. Bross, K. Suhring, y D. Flynn, «HEVC Complexity and Implementation Analysis», *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, n.º 12, pp. 1685-1696, dic. 2012.

[47] G. J. Sullivan, J. Ohm, W. Han, y T. Wiegand, «Overview of the High Efficiency Video Coding (HEVC) Standard», *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, n.º 12, pp. 1649-1668, dic. 2012.

[48] R. A. Olivera Solís, Y. López Pérez, R. A. Olivera Solís, y Y. López Pérez, «Codificación de video en HEVC/H.265 utilizando FFMPEG», *Ing. Electrónica Automática Comun.*, vol. 40, n.º 2, pp. 22-33, ago. 2019.

[49] H. Koumaras, M. Kourtis, y D. Martakos, «Benchmarking the encoding efficiency of H.265/HEVC and H.264/AVC», en *2012 Future Network Mobile Summit (FutureNetw)*, 2012, pp. 1-7.

[50] «Next-Gen Video Encoding: x265 Tackles HEVC/H.265», *Tom's Hardware*, 23-jul-2013. [En línea]. Disponible en: https://www.tomshardware.com/reviews/x265-hevc-encoder,3565.html. [Accedido: 02-may-2019].

[51] «x265 HEVC Encoder / H.265 Video Codec», *x265*. [En línea]. Disponible en: http://x265.org/. [Accedido: 09-abr-2019].

[52] «Encode/H.265 – FFmpeg». [En línea]. Disponible en: https://trac.ffmpeg.org/wiki/Encode/H.265. [Accedido: 02-may-2019].

[53] «Encodear con FFmpeg: HEVC (H.265) • Contempo». [En línea]. Disponible en: https://cont3mpo.github.io/post/2015/encodear-con-ffmpeg-hevc-h265.html. [Accedido: 02-may-2019].

[54] «nginx». [En línea]. Disponible en: https://nginx.org/en/. [Accedido: 04-jun-2019].

[55] «Welcome! - The Apache HTTP Server Project». [En línea]. Disponible en: https://httpd.apache.org/. [Accedido: 02-may-2019].

[56] «Ganglia Monitoring System». .

[57] «FFmpeg». [En línea]. Disponible en: https://ffmpeg.org/. [Accedido: 10-abr-2019].

ANEXOS

Anexo I Conversión de video por *FFmpeg* [57]

FFmpeg *-i entrada.avi input -acodec aac -ab 128kb -vcodec MPEG4 -b 1200kb -mbd 2 -flags +4mv+trell -aic 2 -cmp 2 -subcmp 2 -s 320×180 -title X salida.mp4*

Video original: entrada.avi

Codec de audio: aac

Bitrate del audio: 128kb/s

Codec de vídeo: *MPEG*4

Bitrate del vídeo: 1200kb/s

Tamaño del vídeo: 320×180

Vídeo generado: salida.mp4

Anexo II Opciones Globales del *FFserver*

HTTPPort port_number **Port port_number** **RTSPPort port_number**	**HTTPPort** configura el servidor HTTP que escucha el número de puerto TCP, **RTSPPort** establece el servidor RTSP que escucha el número de puerto TCP. **Port** es el equivalente de HTTPPort y está en desuso. Se debe seleccionar un puerto diferente al del servidor web HTTP estándar si se ejecuta en la misma computadora.

	Si no se especifica, no se creará el servidor correspondiente.
HTTPBindAddress dirección_Ip **BindAddress** dirección_Ip **RTSPBindAddress** dirección_Ip	Establece la dirección en la que está enlazado el servidor HTTP / RTSP. Solo es útil si tiene varias interfaces de red. **BindAddress** es el equivalente de HTTPBindAddress y está en desuso.
MaxHTTPConnections n	Establece el número de conexiones HTTP simultáneas que pueden manejarse. Debe definirse antes del parámetro MaxClients, ya que define el límite máximo de MaxClients. El valor predeterminado es 2000.
MaxClients n	Establece el número de solicitudes simultáneas que se pueden manejar. Dado que ***FFserver*** es muy rápido, es más probable que desee dejarlo tan alto y usar MaxBandwidth. El valor predeterminado es 5.
MaxBandwidth (kbps)	Establece la cantidad máxima de kbit / seg

	que está preparado para consumir al transmitir a los clientes. El valor predeterminado es 1000.
CustomLog	Establece el archivo de registro de acceso (usa el formato de archivo de registro de *Apache* estándar). '-' es la salida estándar. Si no se especifica, ***FFserver*** no generará ningún registro. En caso de que se especifique la opción de línea de comandos -d, esta opción se ignora y el registro se escribe en la salida estándar.
NoDaemon	Esta opción actualmente se ignora, ya que ahora ***FFserver*** siempre funcionará en el modo no-daemon, y está en desuso.
UseDefaults **NoDefaults**	Controla si las opciones de *codec* predeterminadas se usan para todas las secuencias o no. Cada secuencia puede sobrescribir esta configuración por sí misma. El valor predeterminado es *UseDefaults*. La última ocurrencia anula la anterior si existen múltiples definiciones.

Anexo III Archivo de configuración de *FFserver*

```
# Port on which the server is listening. You must select a different
# port from your standard HTTP web server if it is running on the same
# computer.
HTTPPort 8085
RTSPPort 5554

# Address on which the server is bound. Only useful if you have
# several network interfaces.
HTTPBindAddress 0.0.0.0

# Number of simultaneous HTTP connections that can be handled. It has
# to be defined *before* the MaxClients parameter, since it defines the
# MaxClients maximum limit.
MaxHTTPConnections 2000

# Number of simultaneous requests that can be handled. Since FFserver
# is very fast, it is more likely that you will want to leave this high
# and use MaxBandwidth, below.
MaxClients 1000

# This the maximum amount of kbit/sec that you are prepared to
# consume when streaming to clients.
MaxBandwidth 10000

# Access log file (uses standard Apache log file format)
# '-' is the standard output.
CustomLog -

##############################################################################
# Definition of the live feeds. Each live feed contains one video
# and/or audio sequence coming from an FFmpeg encoder or another
# FFserver. This sequence may be encoded simultaneously with several
# codecs at several resolutions.

<Feed feed1.ffm>
File /tmp/feed1.ffm
FileMaxSize 30M
ACL allow 127.0.0.1
ACL allow 10.12.0.0 10.12.255.255
ACL allow 192.168.0.0 192.168.255.255
</Feed>


<Feed feed2.ffm>
File /tmp/feed2.ffm
```

```
FileMaxSize 30M
ACL allow 127.0.0.1
ACL allow 10.12.0.0 10.12.255.255
ACL allow 192.168.0.0 192.168.255.255
</Feed>

<Stream navegador1.webm>                # Output stream URL definition
  Feed feed1.ffm              # Feed from which to receive video
  Format webm

  # Audio settings
  AudioCodec vorbis
  AudioBitRate 64             # Audio bitrate

  # Video settings
  VideoCodec libvpx-vp9       # libvpx
  VideoSize cif               # 720x576         # Video resolution
  VideoFrameRate 15           # 25          # Video FPS
  AVOptionVideo flags +global_header          # Parameters passed to encoder
  AVOptionVideo qmin 1
  AVOptionVideo qmax 31
  AVOptionVideo quality good
  AVOptionAudio flags +global_header
  PreRoll 5
  StartSendOnKey
  VideoBitRate 400            # Video bitrate
  VideoBufferSize 400
</Stream>

<Stream reproductor_265.mp4>                # Output stream URL definition mp4
Feed feed2.ffm
Format mpegts
AudioBitRate 128
AudioChannels 2
AudioSampleRate 44100
VideoBitRate 400
VideoBufferSize 400
VideoFrameRate 15
VideoSize cif
VideoGopSize 23
VideoQMin 1
VideoQMax 31
AudioCodec aac
VideoCodec libx265
AVOptionVideo flags +global_header
AVOptionAudio flags +global_header
PreRoll 5
```

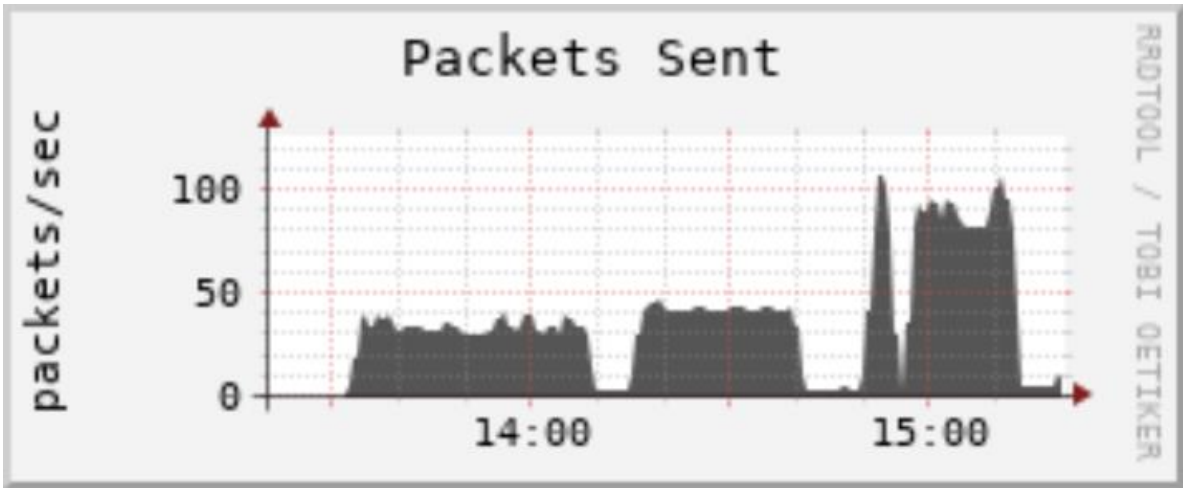
```
</Stream>

<Stream reproductor_264.asf>            # Output stream URL definition
Feed feed2.ffm
Format asf
AudioBitRate 128
AudioChannels 2
AudioSampleRate 44100
VideoBitRate 400
VideoBufferSize 400
VideoFrameRate 15
VideoSize cif
VideoGopSize 18
VideoQMin 1
VideoQMax 31
AudioCodec aac
VideoCodec libx264
AVOptionVideo flags +global_header
AVOptionAudio flags +global_header
PreRoll 5
</Stream>

# Server status

<Stream stat.html>
Format status

# Only allow local people to get the status
ACL allow localhost
ACL allow 10.12.0.0 10.12.255.255
ACL allow 192.168.0.0 192.168.255.255

#FaviconURL http://pond1.gladstonefamily.net:8080/favicon.ico
</Stream>


# Redirect index.html to the appropriate site

<Redirect index.html>
URL http://www.FFmpeg.org/
</Redirect>
```

Anexo IV Archivo de configuración de Nginx

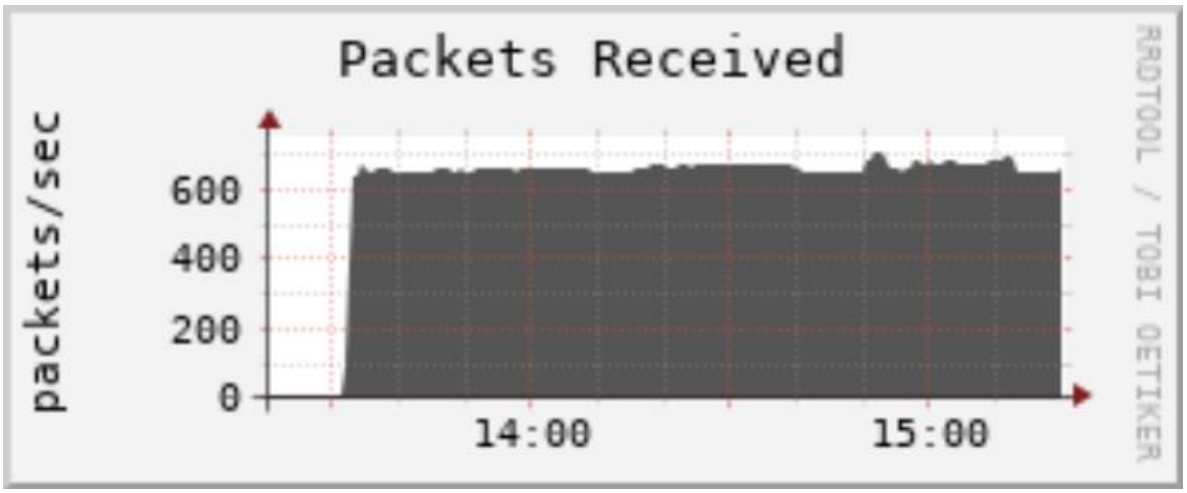
```
worker_processes  3;

events {
    worker_connections  1024;
}


# RTMP configuration
rtmp {
    server {
        listen 1936; # Listen on standard RTMP port
        chunk_size 4000;

application adictos {
        live on;
        record off;
    }

application webcam {
        live on;
         record off;

        # Stream from local webcam
        exec ffmpeg -i rtmp://10.12.xxx.xxx:1936/webcam -crf 30 -preset ultrafast -acodec aac -strict experimental -ar 44100 -ac 2 -b:a 128k -vcodec libx264 -x264-params keyint=60:no-scenecut=1 -r 30 -b:v 800k -maxrate 800k -bufsize 2M -s cif
                    -f flv rtmp://10.12.53.186:1936/webcam/stream;
    }



    }
}


http {
    include       mime.types;
    default_type  application/octet-stream;

    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
```

```
    keepalive_timeout  65;

    #gzIp on;

    server {
        listen       8070;
        server_name  10.12.53.186 ;#www.streaming_hevc.cu

        #charset koi8-r;

        #access_log  logs/host.access.log  main;

        location / {
            root   html;
            index  index.html index.htm;
        }

        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }

    }

}
```

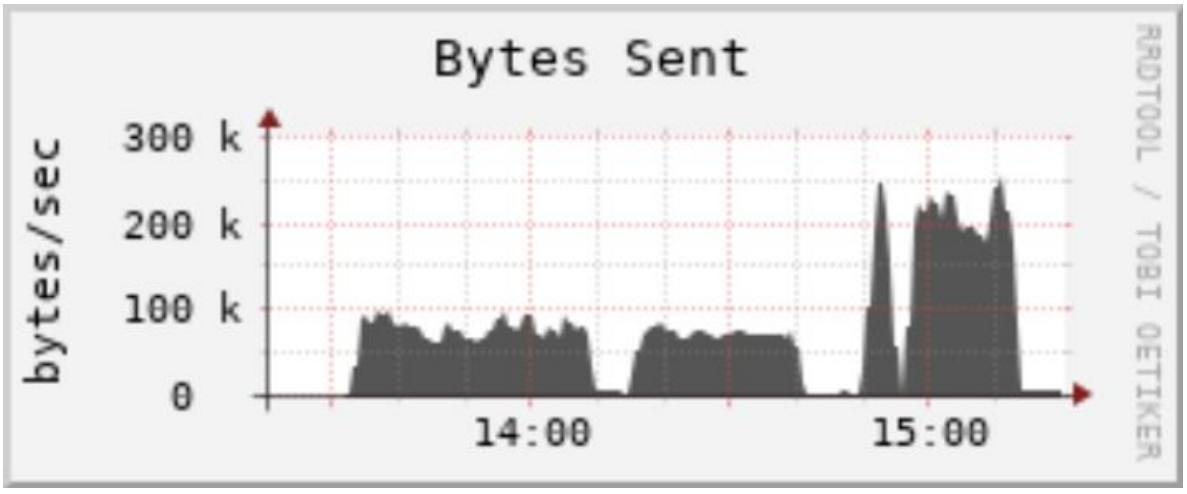
Anexo V Paquetes Enviados



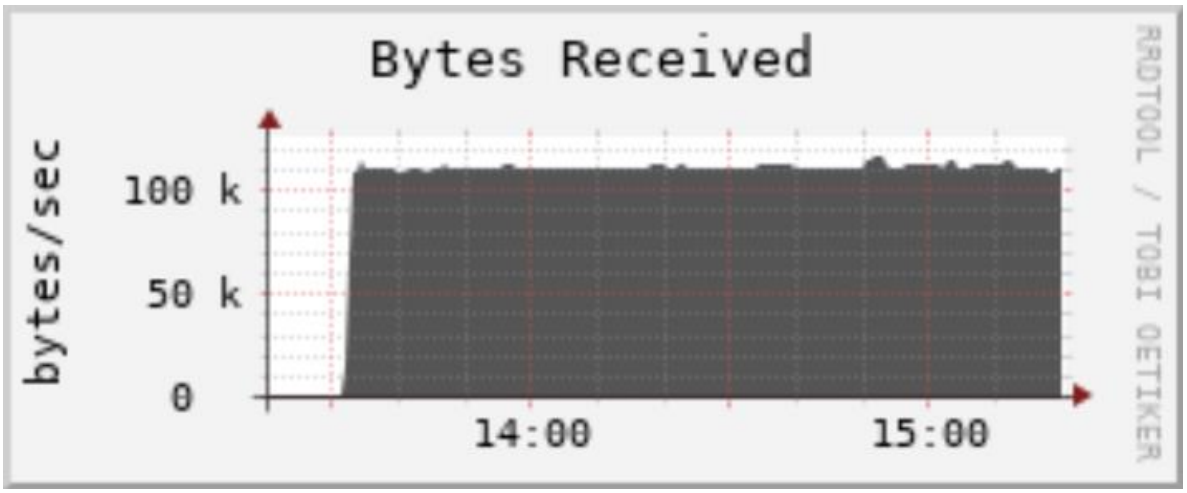
Anexo VI Paquetes Recividos



Anexo VII Bytes enviados



Anexo VIII Bytes recibidos



Anexo IX CPU del sistema