FACULTAD DE INGENIERÍA Y CIENCIAS AGROPECUARIAS

IMPLEMENTACIÓN DE UN PROTOTIPO DE SERVICIO VIDEO STREAMING LIVE PARA LA FICA DE LA UDLA

AUTOR

Javier Alejandro Larrea Sarmiento

AÑO

2017

FACULTAD DE INGENIERÍAS Y CIENCIAS AGROPECUARIAS

IMPLEMENTACIÓN DE UN PROTOTIPO DE SERVICIO VIDEO STREAMING LIVE PARA LA FICA DE LA UDLA

Trabajo de Titulación presentado en conformidad con los requisitos establecidos para optar por el título de Ingeniero en Electrónica y Redes de Información

Profesor Guía

Mg. William Eduardo Villegas Chiliquinga

Autor

Javier Alejandro Larrea Sarmiento

Año

2017

DECLARACIÓN DEL PROFESOR GUÍA

“Declaro haber dirigido este trabajo a través de reuniones periódicas con el estudiante, orientando sus conocimientos y competencias para un eficiente desarrollo del tema escogido y dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación”

_____

William Eduardo Villegas Chiliquinga.
Magister en redes de comunicaciones.
C.I 1715338263

DECLARACIÓN PROFESOR CORRECTOR

Declaro haber revisado este trabajo, dando cumplimiento a todas las disposiciones vigentes que regulan los Trabajos de Titulación".

_____

Luis Santiago Criollo Caizaguano
Master en Redes y Comunicaciones
C.I. 1717112955

DECLARACIÓN DE AUTORÍA DEL ESTUDIANTE

“Declaro que este trabajo es original, de mi autoría, que se han citado las fuentes correspondientes y que en su ejecución se respetaron las disposiciones legales que protegen los derechos de autor vigentes”

_____

Javier Alejandro Larrea Sarmiento

C.I 1726002916

AGRADECIMIENTOS

“Agradezco a mi familia”.

DEDICATORIA

“A las víctimas de la espera”.

RESUMEN

El presente trabajo de titulación tuvo como objetivo el desarrollo e implementación de un prototipo capaz de brindar un servicio de video streaming live para la comunidad universitaria de la FICA-UDLA. Dicha facultad consta de un gran número de estudiantes, y personal docente y administrativo, mismo que se encuentra en constante participación de eventos sociales y culturales que se desarrollan en ambientes limitados de espacio para albergar a toda la comunidad, o a su vez en lugares distantes al establecimiento educativo.

Para cumplir el propósito de este proyecto, se desarrolló un aplicativo Android el cual transmitirá el contenido de video en vivo desde un dispositivo móvil hacia un servidor UNIX. La implementación de un servidor Códec encargado del procesamiento del video y de la codificación en distintos formatos y calidades sobre HTTP, permitió la visualización de video por el usuario usando el aplicativo Wed basado en HTML5, que se desarrolló en el presente estudio. Las pruebas modulares y globales permitieron constatar que existió una correcta recepción y emisión del contenido. Sin embargo, el tiempo de procesamiento introducido por el servidor NGINX empleando el protocolo HLS sobrepasa al tiempo esperado.

Adicionalmente, el software gratuito empleado y el diseño de la implementación se ajustaron a los recursos de hardware empleados, concluyendo que la implementación del prototipo de servicio streaming live para la FICA-UDLA fue factible y eficiente permitiendo mejorar la comunicación entre estudiantes, docentes y empleados en general de la FICA-UDLA mediante la transmisión en tiempo real de eventos desde cualquier navegador web y cualquier sistema operativo que el usuario posea en su ordenador.

ABSTRACT

The current graduating project aimed to develop and to implement a prototype capable of providing a live video streaming service for the university community of FICA-UDLA. This faculty consists of a large number of students, and teaching and administrative staff, who are in constant participation of social and cultural events that take place in limited spaces to accommodate the entire community, or in places far from the Educational establishment.

To fulfill the purpose of this project, an Android application was developed which will transmit live video content from a mobile device to a UNIX server. The implementation of a codec server in charge of video processing, and encoding in different formats and qualities over HTTP, allowed the visualization of video by the user using the HTML5 based Wed application developed in the present study. The modular and global tests allowed to verify that there was a correct reception and emission of the content. However, the processing time introduced by the NGINX server using the HLS protocol exceeds the expected time.

In addition, the free software used in this project and the design of the implementation were adjusted to the hardware resources employed, concluding that the implementation of the live streaming service prototype for the FICA-UDLA was feasible and efficient, allowing a better communication between students, teachers and employees in general of the FICA-UDLA through the real-time transmission of events from any web browser and any operating system that the user owns in their computer.

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Alcance

El alcance de este trabajo de titulación se limita a la implementación del prototipo de video streaming live en el data center de la FICA de la UDLA.
Los componentes del prototipo a desarrollar son:

Aplicación Móvil. - Este software estará orientado para usuarios del Sistema Operativo Android el cual permitirá a los miembros de la comunidad universitaria captar video a través de la cámara incorporada en sus teléfonos celulares y emitirlo hacia el servidor que se encargará de la multidifusión del mismo.

Servidor Códec. - Este componente será el encargado de decodificar y codificar el streaming de video recibido del aplicativo Android hacia formatos de multidifusión de video sobre HTTP.

Aplicación Web. - Se desarrollará un aplicativo Web basado en HTML5 el cual permitirá a los usuarios finales visualizar el contenido de video en vivo mediante un explorador web.

El prototipo trabajará en conjunto usando como medio tanto la red cableado como la inalámbrica que posee la FICA de la UDLA.

Justificación

La Universidad de las Américas compuesta por la comunidad estudiantil, administrativa y docente se encuentra en constante participación de eventos sociales, culturales y de interés colectivo mismos que son realizados en las distintas sedes de la universidad.

La UDLA consta de una robusta infraestructura de red tanto cableada como

inalámbrica capaz de soportar altas cargas de tráfico, la cual puede ser accedida desde cualquier lugar de la universidad.

La iniciativa nace de la dificultad que tiene la comunidad universitaria para movilizarse de una sede a otra o en su defecto la capacidad de los lugares en que los eventos masivos son realizados restringiendo el ingreso completo de las personas interesadas en dichos eventos, por estos motivos se propone la implementación de un servidor de Streaming para la FICA mismo que se encargará de la difusión en tiempo real del evento de interés y podrá ser accedido desde cualquier lugar que conforme la UDLA, de esta manera se pretende mejorar la experiencia de los eventos masivos para la comunidad.

Objetivos

Objetivo General

Diseñar e implementar el servicio de video streaming live para la FICA de la UDLA a manera de prototipo, por medio de un servidor de multidifusión y una aplicación celular para la generación de contenido.

Objetivos Específicos

- Analizar y seleccionar los protocolos que más se ajusten a las necesidades para esta implementación de video Streaming live.
- Implementar un servidor virtualizado encargado de la difusión del contenido accesible mediante un aplicativo web.
- Desarrollar un aplicativo Android encargado del envío de contenido en tiempo real hacia el servidor de multidifusión.
- Realizar pruebas de funcionamiento global del prototipo.

1. CAPÍTULO I. MARCO TEÓRICO

La distribución y difusión de contenidos multimedia de manera continua y en tiempo real a través del internet es una herramienta de comunicación barata y sencilla en su manejo, por lo que el uso de tecnologías como streaming, el webcasting y postcasting son usados hoy en día por investigadores, profesionales, científicos y público en general (Alonso, Navarro, López, González de los Ríos y Aleizandre, 2015, pp. 278-283).

1.1 Streaming

La tecnología streaming se define como la transmisión continua de datos desde un servidor multicast a través de una red hacia varios clientes, fue desarrollada con el objetivo de poder consumir contenido multimedia desde un servidor sin la necesidad de descargarlo completamente en el cliente. Es decir, podemos visualizar el contenido mientras este es descargado en nuestro dispositivo (Du, Liu, Liu y Chen, 2014, pp. 275-279).

Actualmente existen tres únicas maneras de transmitir video a través de Internet: streaming tradicional, descarga progresiva y streaming adaptivo.

1.1.1 Streaming Tradicional (File Download)

Se genera una conexión entre servidor y cliente con la cual mutuamente conocen sus estados, como consecuencia el servidor envía paquetes hasta que el buffer del cliente se encuentre lleno, además de esto el bit-rate con el que el video es codificado y transmitido se mantiene constante durante toda la conexión, es decir la tasa de codificación deberá ser menor que el ancho de banda disponible para evitar interrupciones durante la transmisión del video (Du et al, 2014, pp. 275-279).

En esta variedad de streaming, el usuario deberá esperar que todo el contenido sea descargado en el dispositivo para poder ser visualizado.

1.1.2 Descarga Progresiva

Se basa en solicitudes HTTP donde el contenido del video es descargado paulatinamente en el dispositivo cliente y este puede ser visualizado paralelamente a esta actividad, el contenido seguirá siendo descargado, aunque el video se encuentre pausado por el usuario, cuando la totalidad del contenido es descargado este se encontrará almacenado en caché y se podrá visualizar cualquier tramo del contenido.

1.1.3 Streaming Adaptivo

Generalmente es utilizado en conjunto con HTTP donde el contenido es transmitido al cliente de manera segmentada diferenciando en los recursos disponibles entre el cliente y el servidor, a diferencia del método de descarga progresiva donde se transmite el contenido completo en una sola codificación y sin ninguna distinción en cuanto a la calidad del video que está siendo descargado.

En este método, cada segmento nuevo (chunk) a ser transmitido es independiente del anterior en lo que respecta a la calidad del video que este contiene y la adaptación del bit-rate que se encuentra codificada en cada segmento generalmente es dependiente del ancho de banda disponible entre el cliente y el servidor, es decir entre mayor ancho de banda disponible en el momento de la solicitud del streaming, la calidad del video será mejor en función de los bit-rate disponibles para este contenido en el servidor (Seufert, Egger, Slanina, Zinner, Hoßfeld y Tran, 2015, pp. 469-492).

En la siguiente figura se puede apreciar claramente el proceso en el cual los clientes consumen los recursos de video a una mayor o menor resolución basado en el ancho de banda disponible entre los clientes y el servidor.

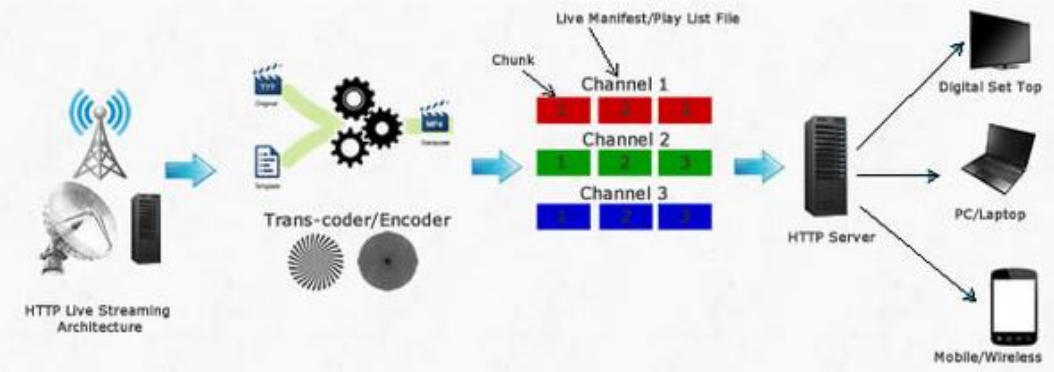


Figura 1. Streaming adaptivo de video.
Tomado de Vayavya Labs Pvt. Ltd, 2014.

1.2 Tipos de servicio Streaming

Existen dos tipos de servicio streaming que se pueden ofrecer a los usuarios, los cuales han surgido de las distintas necesidades de negocio a las que se orientan las empresas que se desarrollan en este entorno.

1.2.1 En Directo (Live Streaming)

Es la transmisión en tiempo real de un evento que está ocurriendo ese instante, es decir, el contenido solo puede ser visualizado en el momento de la transmisión del streaming ya que el contenido va siendo eliminado conforme es receptado y visualizado por el usuario final, este comportamiento se asemeja al de un televisor convencional (Thang, Le, Pham y Ro, 2015, pp. 189-193).

1.2.2 Bajo Demanda (On-Demand)

Se puede descargar el contenido que se encuentra almacenado en un servidor remoto para ser visualizado en cualquier momento mientras se tenga disponible una conexión a internet, un claro ejemplo de esto es YouTube y Netflix los cuales ofrecen una gran variedad de contenido de videos y películas accesibles desde un navegador web (Thang et al, 2015, pp. 189-193).

1.3 Componentes básicos de una arquitectura de video streaming

En una arquitectura de video streaming estándar la comunicación entre el cliente y el servidor utiliza el protocolo RTSP (Real Time Streaming Protocol) o cualquiera de sus derivados, para ofrecer características como salto, adelanto de la visualización del contenido en el caso del video sobre demanda.

Una solución de video Streaming pasa por varios procesos a través de los componentes de la arquitectura hasta llegar al usuario final para que el contenido pueda ser visualizado. De manera general la arquitectura de video Streaming está compuesta por el equipo de producción, equipo de almacenamiento, servidor de codificación y streaming, red y aplicación cliente (Hoque, Siekkinen, Nurminen, Tarkoma y Aalto, 2017, pp. 1-23 y Thang et al, 2015, pp 189-193).

- Equipo de producción. - Es el conjunto de hardware y software encargado de la captura de contenido de audio y video a través de una cámara, micrófono u otro medio para su posterior edición.
- Equipo de almacenamiento. - Es el hardware en el cual la información es almacenada para poder ser accedida por el cliente, este puede ser un componente del propio servidor o en su defecto estar reservado lógicamente en un almacenamiento externo.
- Servidor de codificación y streaming. - Es el encargado de la recepción del contenido y la conversión a un formato fácilmente manipulable para distribuirlo hacia las aplicaciones cliente.
- Red. - Es el recurso usado como medio de comunicación entre el cliente y el servidor para poder transmitir datos.
- Aplicación cliente. - Encargada de la recepción y presentación del contenido al usuario final, generalmente el uso de un browser de páginas web es la mejor opción debido a su flexibilidad, escalabilidad y aceptación global.

1.4 Protocolos de streaming

Un protocolo de comunicación es un conjunto de reglas y parámetros que son definidos con la finalidad de crear un estándar común para todos los proveedores o fabricantes de hardware y software.

Sea cual sea la arquitectura de video streaming que se esté implementando debe estar al margen de los protocolos definidos globalmente, para facilitar la interacción con los usuarios, los principales protocolos relacionados con el video streaming son (Hoßfeld, Schatz y Krieger, 2014, pp. 136-150 y Hoque et al, 2017, pp. 1-23).

1.4.1 User Datagram Protocol (UDP)

Es un protocolo no orientado a la conexión que es parte de la capa de transporte del modelo TCP/IP, provee escasos recursos para la identificación y corrección de errores durante la transmisión, es altamente eficiente en aplicaciones que son susceptibles al tiempo y que no requieren de una entrega confiable tal como llamadas IP, video bajo demanda, etc. Además, no se requiere a procesamiento o traducción adicional entre la fuente de datos y la tabla de salida de datos (Renard, K. y Adametz, J., 2015)

1.4.2 Protocolo de Control de Transmisión (TCP)

Es un protocolo orientado a la conexión el cual provee varios mecanismos de detección de errores, retransmisión de paquetes y control de flujo, debido a su complejidad es mucho más lento respecto al protocolo UDP, pero en su lugar provee confiabilidad en la entrega de paquetes de manera correcta (Renard, K. y Adametz, J., 2015).

1.4.3 Protocolo de Transporte en tiempo Real (RTP)

Está basado en el protocolo UDP, generalmente es usado para la transferencia de audio y video, pero no posee control de flujo por lo cual su uso en solitario no es recomendado (Túqueres, C., 2015).

1.4.4 Protocolo de Control en Tiempo Real (RTCP)

Está basado en UDP y es el encargado de gestionar el control de flujo de RTP (Túqueres, C., 2015).

1.4.5 Protocolo de Mensajes en Tiempo Real (RTMP)

Está basado en TCP y es usado por programas de Flash media Player para comunicarse con servidores de audio y video, permite la transmisión de contenido y el control tanto de la calidad de la transmisión como de la seguridad de este, es propietario de Adobe (Pattaranantakul, M., Sanguannam, K., Sangwongngam, P. y Vorakulpipat, C., 2015)

1.4.6 Protocolo de Transferencia de Hipertexto (HTTP)

Es el estándar en lo que se refiere a Web, permite la solicitud de ficheros específicos alojados en el servidor, como consecuencia se puede obtener usa sección concreta del streaming de video para ser reproducido en el navegador del cliente (Begen, A., Akgul, T. y Baugher, M., 2011).

1.4.7 Streaming en Vivo sobre HTTP (HLS)

Es un protocolo basado en HTTP como su nombre lo indica el cual permite la visualización de streaming de video adaptivo a través de reproductores web independientemente de la plataforma o dispositivo cliente, su propietario es Apple pero es altamente usado debido a su soporte multiplataforma (Begen, A.et al., 2011).

1.4.8 Moving Picture Experts Group-Dynamic Adaptive Streaming over HTTP (MPEG-DASH)

Al igual que el protocolo HLS permite la visualización de video streaming de forma adaptiva, está pensado para convertirse en un estándar global eliminando las distintas encapsulaciones que existen en cada proveedor y

brindar un protocolo de uso común para compatibilidad de aplicaciones de streaming sobre HTTP (Thomas, E., Van, M., Begen, A., Famaey, J. y Stockhammer, T., 2015).

1.5 Calidad del video

Comúnmente se tiene la idea errónea de que la calidad de contenido esta ligada únicamente de la resolución, si hacemos referencia a la calidad de video esta depende de varios parámetros que deben ser tomados en cuenta con mayor razón si el contenido está siendo emitido en tiempo real, de manera sencilla la calidad de video se puede expresar de la siguiente manera (García, Cabrera y García, 2014, pp. 574-575 y Hoßfeld et al, 2014, pp. 136-150).

Alta resolución + Alto BitRate + Códec eficiente = Alta calidad de video (Ecuación 1)

1.5.1 Resolución

Es el grado de detalle o nitidez, la resolución esta ligada a una relación entre la dimensión de la imagen, video o medio, versus el número de pixeles que esta puede presentar, por esta razón la resolución es medida en número de pixeles o puntos por pulgada lineal, tomando como referencia la resolución de una imagen entre mayor número de pixeles por pulgada lineal esta pueda representar, mayor resolución será obtenida (García et al, 2014, pp. 574-575). La figura 2 ejemplifica la resolución que puede ser representada basada en el número de pixeles, más pixeles es sinónimo de mayor resolución, aplicable tanto para contenido multimedia gráfico como para dispositivos de reproducción.

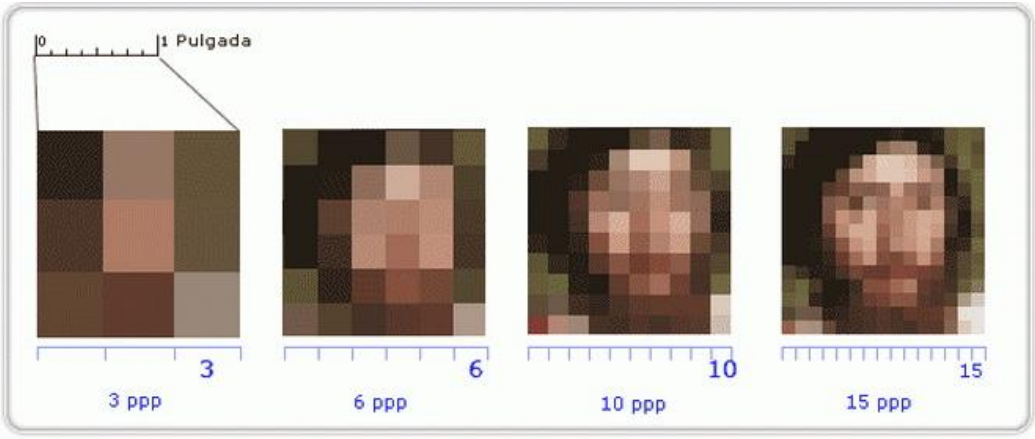


Figura 2. Pixeles versus resolución
Tomado de Adro4all, 2013.

1.5.2 Tasa de bits (BitRate)

Es la media de datos o bits que son insertados en una imagen o video en el intervalo de un segundo, es decir, es un factor que toma mucha importancia si se requiriere una buena calidad del video la cual está relacionada con la resolución, es decir una imagen de alta resolución transmitida a un bajo BitRate tendrá como resultado una baja calidad de la misma, por lo tanto, será necesario ajustar la tasa de bits según la resolución a ser transmitida. La figura 3 representa los resultados de visualización ante una alta y baja tasa de BitRate.

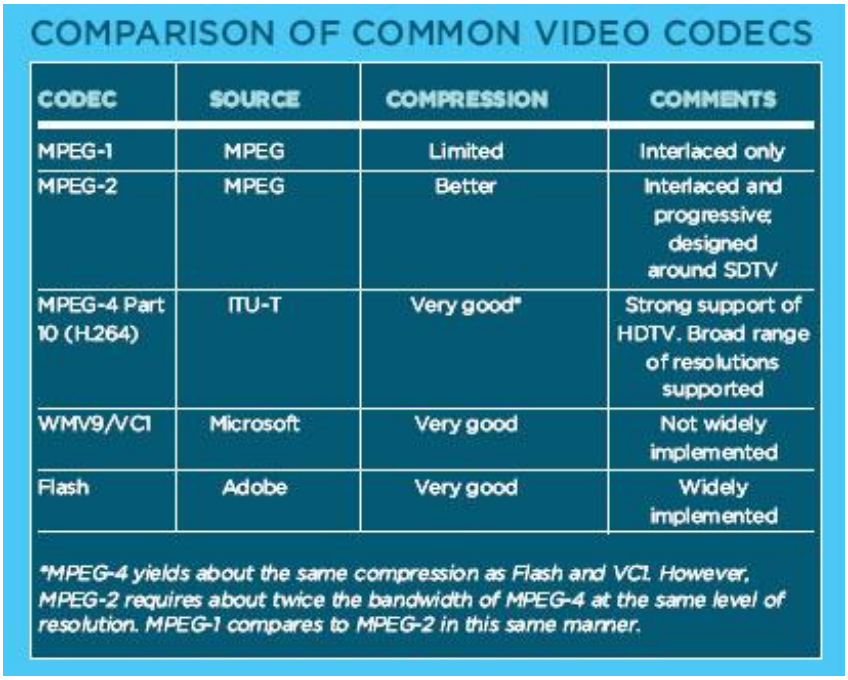
Figura 3. Variación de BitRate en la visualización de contenido
Tomado de Mente Curiosa.net, 2017

1.5.3 Códec

El término Códec se atribuye a la derivación abreviada de los términos Codificador/Decodificador, un Códec puede ser ejecutado tanto en hardware como en software y es utilizado para comprimir audio, imagen o video con la finalidad de reducir la cantidad de almacenamiento que ocupa el contenido multimedia, en las redes promedio actuales es necesario el uso de estos mecanismos de compresión debido a la carga excesiva que genera la transmisión de este tipo de contenidos. Solamente existen dos formas de comprimir el contenido multimedia con ayuda de códecs, estas son con pérdidas (lossless) y con pérdidas lossy (Li y Liang, 2217, pp. 1-35).

- Lossless. - Genera un contenido mucho más pesado que la otra variedad de códecs debido a que diseñado para no recortar u omitir información, es decir los datos antes del proceso de codificación serán exactamente los mismo después de la descompresión.
- Lossy. - Omite contenido que podría ser imperceptible o es repetitivo en una secuencia de datos, ejemplificando esta consideración al momento de emitir un video los algoritmos de codificación toman lugar analizando los fotogramas de la secuencia y omitiendo fragmentos que se repiten entre ellos.

De esta manera se puede afirmar que los códecs son de vital importancia en la visualización de contenido, ya que estos influencian directamente en el proceso de transmisión de los mismos, pudiendo aligerar la carga y retardo que se presenta sobre el medio de transmisión. La figura 4 presenta una comparación de eficiencia entre los principales códec de videos existentes actualmente.



COMPARISON OF COMMON VIDEO CODECS

CODEC	SOURCE	COMPRESSION	COMMENTS
MPEG-1	MPEG	Limited	Interlaced only
MPEG-2	MPEG	Better	Interlaced and progressive; designed around SDTV
MPEG-4 Part 10 (H.264)	ITU-T	Very good*	Strong support of HDTV. Broad range of resolutions supported
WMV9/VC1	Microsoft	Very good	Not widely implemented
Flash	Adobe	Very good	Widely implemented

**MPEG-4 yields about the same compression as Flash and VC1. However, MPEG-2 requires about twice the bandwidth of MPEG-4 at the same level of resolution. MPEG-1 compares to MPEG-2 in this same manner.*

Figura 4. Comparativa de códecs de video
Tomada de AVNetwork.com, 2008.

1.6 Parámetros de comunicación para transmisión de contenido multimedia

En la actualidad la mayoría de redes LAN están basadas en el protocolo IP el cual por defecto distribuye los recursos de la red de una manera equitativa, esto puede resultar perjudicial para las comunicaciones susceptibles al tiempo como lo es el streaming de video, los problemas que suelen presentarse comúnmente son debido a dos factores:

- Una red basada en con mutación por paquetes como lo es IP siempre elegirá el mejor camino lo que provocará el aparecimiento de jitter.
- Ya que las aplicaciones multimedia en una red son perceptibles al tiempo un retraso puede ser muy perjudicial para la transmisión del contenido.

Existen varios factores maliciosos para la transmisión de contenido los cuales son consecuencia directa del desempeño de la red y su funcionamiento.

1.6.1 Retardo o Latencia

En redes se define al retardo como el tiempo transcurrido desde la emisión del contenido multimedia hasta su recepción por parte del cliente, es decir el tiempo transcurrido para llegar de un extremo a otro atravesando la red. Existen varios tipos de retardos provocados por limitaciones de hardware o por el medio en que las transmisiones se realizan (Bouzakaria, Concolato y Le, 2014, pp. 92-97).

- Retardo de procesamiento. - Es el retardo introducido por los equipos de red al procesar los paquetes o por los servidores y clientes para codificar y decodificar el contenido multimedia.
- Retardo de señalización. - Es el tiempo que se tardan los equipos de red en codificar el contenido multimedia en un formato común y entendible para el medio de transmisión.
- Retardo de propagación. - Es el tiempo que tardan los paquetes para llegar de un equipo de transmisión a otro por un cierto medio.
- Retardo de encolado. - Es el tiempo en buffer de un equipo de red que debe esperar un paquete antes de ser transmitido, esto ocurre generalmente cuando existe saturación de enlaces.

La figura 5 representa un caso ejemplo de la latencia inducida en una trasmisión de eventos.

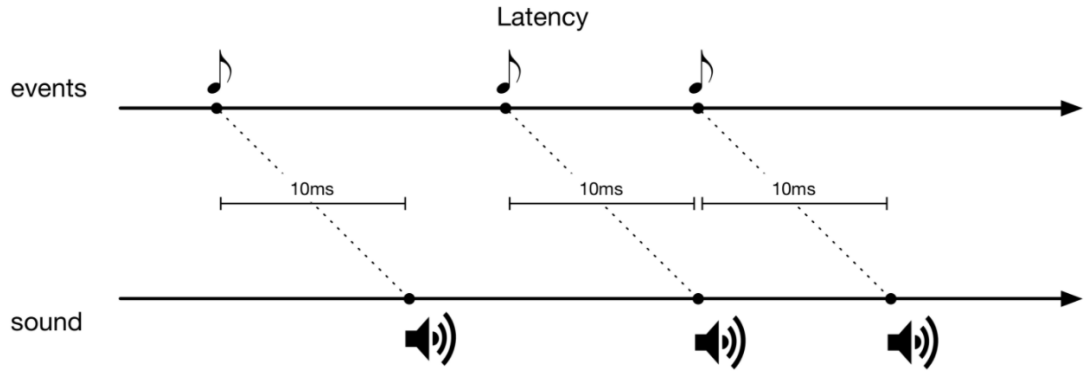


Figura 5. Representación de Latencia
Tomado de Expressiveness, 2012

1.6.2 Jitter

Aunque existen varias definiciones para el Jitter la más acertada lo determina como la variación de retardo o latencia durante una transmisión, como ya se ha mencionado las redes conmutadas por paquetes no nos aseguran que la secuencia de emisión de los paquetes desde un origen sea la misma de recepción en el destino, generalmente esto ocurre debido a los algoritmos que este tipo de redes emplean donde no se da lugar a un camino determinado para todo el flujo de paquetes de la comunicación, sino más bien estos son distribuidos por las posibles rutas disponibles en la red que en ese momento son los idóneos para la conmutación lo que conlleva a una variación en el tiempo de llegada de los paquetes (Liang y Liang, 2008, pp. 1520-9210)

El Jitter puede acarrear varios problemas con la experiencia del usuario cuando datos susceptibles al tiempo están involucrados ya que los equipos finales tienden a descartar los paquetes receptados cuando estos no son recibidos cuando son esperados, esto se ve reflejado como interrupciones en la comunicación y o una voz robotizada (Sanogo, 2010, pp. 1-18).

Los valores recomendados para el jitter deben ser menores a 100 ms, en el caso de que no sea posible cumplir con los valores recomendados la mejor opción es la implementación de un Buffer tanto en clientes como en servidores según sea necesario, esta solución aumenta retraso en la transmisión, pero asegura una entrega de mayoría de los paquetes de una manera ordenada. La figura 6 representa la variación de retardo de la señales para una comunicación en particular.

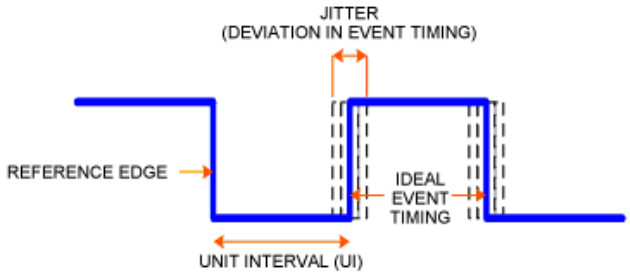


Figura 6. Representación de Jitter
Tomada de Sanogo, 2010, pp. 1-18.

1.6.3 Ancho de Banda

El ancho de banda es considerado como la cantidad de información que puede ser transmitida por un medio o canal en un intervalo de tiempo, comúnmente se encuentra expresado en megabytes por segundo (Mbps) o gigabytes por segundo (Gbps), como norma general el ancho de banda disponible debe ser mayor al ancho de banda requerido para el envío (Hoßfeld et al, 2014, pp. 136-150).

El ancho de banda es un recurso limitado en todas las redes por lo cual es necesario el uso pertinente de este, el enviar tráfico superior al ancho de banda disponible en los enlaces de los equipos intermedios podrían provocan repercusiones en la red. Cuando no existe ancho de banda disponible para transmitir el conjunto de datos deseado el comportamiento por defecto de los equipos intermedios de red consiste en descartar paquetes ya sea diferenciando sus prioridades o aleatoriamente.

1.7 Métodos de difusión de contenido

Para el correcto desempeño de la red se han diseñado distintos tipos de difusión de información basados en el destino al que se quiere transmitir, es decir no se enviará datos a todos los miembros de la red a menos que esto se requerido, estos métodos se basan en la capa 2 del modelo OSI y se clasifican de la siguiente manera.

1.7.1 Unicast

Consiste en el envío de información desde un único origen hacia un único destino, aunque la información a ser enviada en un cierto instante de tiempo coincida para dos dispositivos finales se generará un flujo de datos por cada dispositivo que requiere esta información, en otras palabras, si “n” dispositivos requieren simultáneamente los mismos datos de un único servidor este generará “n” respuestas, una por cada petición (Keshavarz y Riedi, 2014, pp. 102-115).

1.7.2 Multicast

Es un método de difusión altamente eficiente cuando se requiere transmitir simultáneamente los mismos datos desde un origen común hacia varios destinos, pero no a todos los miembros de la red, debido a que solamente existe una copia del flujo de datos esta se multiplica o distribuye en los dispositivos intermediarios de la red cuando es necesario hasta llegar a los dispositivos finales ahorrando un considerable ancho de banda. La figura 7 a continuación señala la relación del consumo de banda relacionando el tráfico únicas versus multicast.

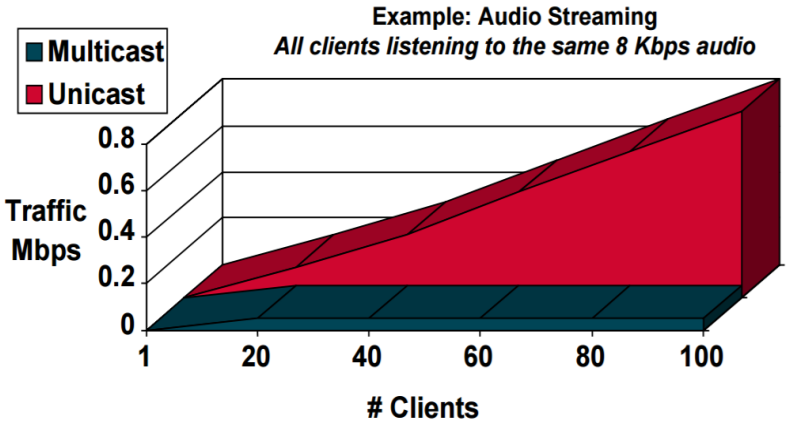


Figura 7. Comparación del tráfico multicast Vs. unicast en redes IP. Tomado de CISCO, 2000.

1.7.3 Broadcast

Se basa en una única copia del flujo de datos que es enviada desde un origen común y es distribuida por los dispositivos intermedios de red hacia todos los miembros de la red, el conjunto de todos dispositivos que conforman el destino de esta difusión es conocido como dominio de Broadcast, el cual está limitado por un dispositivo de capa tres del modelo OSI y sus miembros están definidos mediante el protocolo ARP (Keshavarz y Riedi, 2014, pp. 102-115).

Un claro ejemplo de este método de difusión es la señal de radio misma que es emitida a todos los receptores incluso si al usuario final no le interesa su contenido.

1.8 Android

Es un sistema operativo basado en Linux el cual fue creado inicialmente para funcionar sobre teléfonos móviles, Android fue desarrollado por un consorcio de 48 empresas, pero liderados principalmente por Google, la finalidad de este sistema operativo es proporcionar un sistema estable, multiplataforma, libre y eficiente que pueda aprovechar todas las capacidades disponibles de un teléfono inteligente. A continuación, se detallan algunas características propias de este sistema operativo (Backes, Bugiel, Derr, McDaniel, Octeau y Weisgerber, 2016, pp. 1-19).

- Soporta una gran cantidad de formatos de audio y gráficos.
- Es multiplataforma, capaz de funcionar en Smartphones, relojes, tablets, etc.
- Al estar basado en Linux nos proporciona un entorno de desarrollo abierto, es decir se puede usar y modificar sin el pago de una licencia.
- Sus aplicaciones están basadas en su gran mayoría en Java lo cual nos asegura la portabilidad de las mismas gracias al concepto de máquina virtual.
- Cada aplicación requiere sus permisos específicos los cuales son aprobados previamente por el usuario del dispositivo, limitando de esta manera la interacción entre las mismas y brindado un medio de seguridad al dispositivo y la información contenida en el mismo.

1.8.1 Arquitectura de Android

La arquitectura de Android está compuesta por capas, una capa superior consume los recursos de las capas que se encuentran debajo de esta, a manera de ejemplo las librerías nos permiten la interacción con los componentes de hardware sin la necesidad de usar programación de bajo nivel para interactuar con estos. A continuación la figura 8 representa gráficamente la arquitectura base del sistema operativo Android.

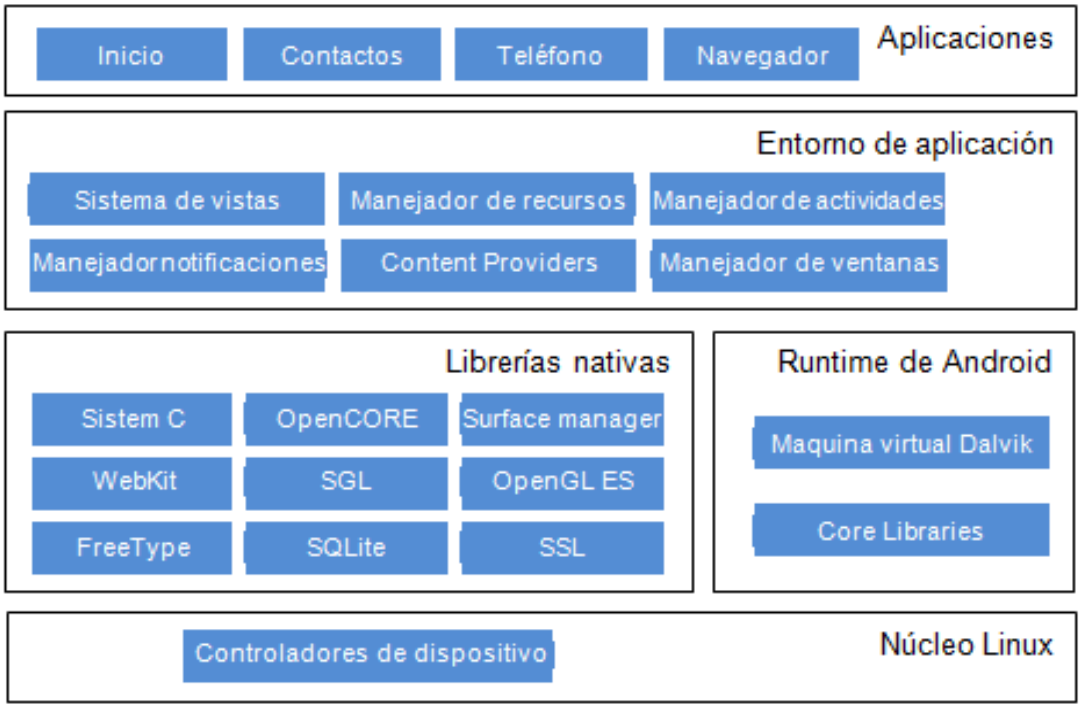


Figura 8. Sofware Android con flujo de datos
Tomado de Backes et al, 2016.

1.8.1.1 Núcleo Linux

Es la capa base de interacción inmediata con el Hardware del dispositivo, el núcleo de Android está formado a partir del sistema operativo Linux 2.6. Esta capa está encargada de gran variedad de funciones derivadas de UNIX las cuales incluyen gestión de memoria y procesamiento, funciones de seguridad, etc. (Kumar, 2014, pp. 33-43 y Backes et al, 2016, pp. 1-19).

Esta capa contiene todos los drivers necesarios para la interacción con los componentes del dispositivo, es decir que al añadir o modificar un componente o driver controlador se producirá una alteración en el núcleo Linux de Android.

1.8.1.2 Librerías nativas

Las librerías nativas están escritas en C y C++ y están compiladas en código nativo del procesador, estas librerías proporcionan las características más representativas del sistema operativo Android.

1.8.1.3 Runtime de Android

Está basada en el concepto de máquina virtual de Java, al considerar los ambientes con baja memoria y procesamiento en los cuales correría esta máquina virtual se decidió optar por un nuevo diseño y desarrollo que dio como resultado la creación de Dalvik que posteriormente sería reemplazada por ART (incluida en versiones Android 5.0 o posteriores) misma que reduce hasta en un treinta y tres por ciento el tiempo de ejecución del código Java de su antecesora.

Cada aplicación corre su propia instancia de máquina virtual ejecutando archivos DEX el cual es un formato creado especialmente para las máquinas virtuales de Android, además Dalvik y ART están basadas en registros y delegan ciertas funciones y procesamientos de bajo nivel al núcleo de Linux para optimizar la operación de la arquitectura (Kumar, 2014, pp. 33-43 y Yadav y Bhadoria, 2015, pp. 1076-1079)

1.8.1.4 Entorno de aplicación

Extensamente conocido también como Application Framework, ofrece el conjunto global de funciones para el desarrollo de aplicaciones las cuales han sido desarrolladas ya sea por Google o por personas externas debido a la condición de código abierto en el que este sistema operativo se encuentra. El entorno de aplicación contiene los servicios necesarios para que una aplicación sea ejecutada y desplegada al usuario, los servicios más relevantes son:

- Sistema de vistas. - Como su nombre lo indica se encarga de la parte visual de los aplicativos.
- Manejador de recursos. - Brinda acceso a recursos externos que no sean código, tales como audio, imágenes, archivos de diseño, etc.
- Manejador de actividades. - Proporciona un sistema de navegación entre aplicaciones y controla el ciclo de vida de las mismas.
- Manejador de notificaciones. - Usa la barra de estados como medio para que las aplicaciones puedan mostrar sus notificaciones

personalizadas.

- Content Providers. - Provee un mecanismo para el acceso de datos de otras aplicaciones.
- Manejador de Ventanas. - hace uso de la librería Surface Manager para gestionar las distintas ventanas de las aplicaciones.

1.8.1.5 Aplicaciones

Es la capa superior de la arquitectura Android, en donde se alojan tanto las aplicaciones propias del sistema como las desarrolladas por terceros, es decir a manera de ejemplo una aplicación de mensajería SMS del sistema puede ser reemplazada de manera acertada por una aplicación desarrollada que ofrece las mismas funcionalidades pero está desarrollada por un tercero, cada aplicación utiliza una instancia de máquina virtual para proveer seguridad entre aplicaciones (Yadav y Bhadoria, 2015, pp. 1076-1079).

1.8.2 Componentes de una aplicación Android

La arquitectura de Android está orientada hacia la reutilización de recursos por tanto cualquier aplicación puede publicar sus capacidades para ser utilizadas por otra, cada componente es una entidad individual aunque puedan existir dependencias entre ellos, para ejemplificar este comportamiento supondremos el desarrollo de una aplicación que hace uso de la aplicación de cámara, las medios generados por el uso de la cámara podrán ser guardados en la aplicación galería donde posteriormente podrán ser editados, las aplicaciones Android hacen uso de los siguientes componentes (Kumar, 2014, pp. 33-43 y Yadav y Bhadoria, 2015, pp. 1076-1079):

1.8.2.1 Actividades

Se define como el conjunto de pantallas gráficas que ofrecen una interacción con el usuario, aunque cada actividad o pantalla es independiente de las otras

estas trabajan en conjunto para ofrecer al usuario una mejor experiencia, por ejemplo, un directorio de contactos ofrece una actividad que muestra el conjunto de nombres registrados, al dar un toque en cualquiera de estos nombres se nos desplegará otra actividad con los detalles del contacto elegido (Kumar, 2014, pp. 33-43).

1.8.2.2 Servicios

Son tareas que se ejecutan en segundo plano, es decir no están en interacción directa con el usuario por tanto no ofrecen una interfaz gráfica, los servicios son iniciados y finalizados por los otros componentes de Android cuanto esto es necesario.

1.8.2.3 Proveedores de contenido

Hace referencia a las ubicaciones de almacenamiento persistente como lo son las bases de datos las cuales están disponibles para nuestras aplicaciones como es el caso de las agendas cuyos datos están disponibles para todas las aplicaciones que los requieran y pueden ser modificados de ser necesario.

1.8.2.4 Receptores de mensajes

Es un componente que se encarga de escuchar y orquestar los mensajes de las aplicaciones y el sistema en general, por tal motivo no esta provisto de una interfaz gráfica, aunque puede mostrar notificaciones al usuario por medio de la barra de estado, aunque por lo general se encarga de anunciar los recursos disponibles a las aplicaciones interesadas e iniciar y detener servicios (Yadav y Bhadoria, 2015, pp. 1076-1079).

2. CAPÍTULO II. ANÁLISIS DE REQUERIMIENTOS, ELECCIÓN DE PROTOCLOS Y SOFTWARE.

Este capítulo tiene como objetivo analizar los distintos requerimientos que deberá cumplir el prototipo en base a las respuestas obtenidas de los usuarios que harán uso del mismo, adicionalmente se realizará la elección de protocolos y software necesario para cumplir con la meta del prototipo.

Se ha realizado la recopilación de los requisitos que deberá tener el sistema, así como también la factibilidad de desarrollar del mismo. Esta información ha sido obtenida mediante encuestas dirigidas a una muestra de treinta y dos miembros de la comunidad estudiantil de la FICA de la UDLA elegidos aleatoriamente.

2.1 Factibilidad del prototipo

Es necesario definir si el prototipo a desarrollar será de utilidad para el grupo al cual está dirigido. Para definir la factibilidad de desarrollar el prototipo se ha planteado una pregunta dirigida a los encuestados relacionada con el uso en futuro el contenido de video streaming (Ver Figura 9).

1. ¿Usaría usted este sistema ya sea como emisor o receptor del contenido de video?

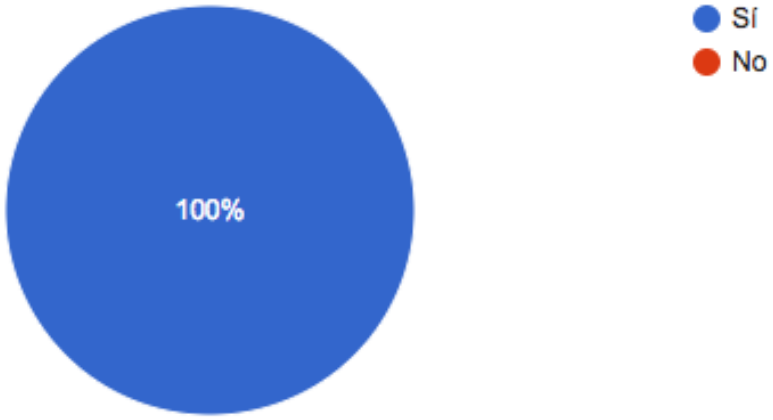


Figura 9. Factibilidad del desarrollo del prototipo video streaming live en la UDLA.

Tal como se puede observar en el gráfico estadístico procesado, el cien por ciento de los encuestados está de acuerdo con el desarrollo del prototipo y consideraría hacer uso del mismo ya sea como generador de contenido o espectador.

2.2 Requerimientos del prototipo

Para un correcto diseño de la solución es necesario conocer los requerimientos que debe cumplir el prototipo y saber diferenciar entre funcionales y no funcionales, ya que tal como se indicó en el segundo capítulo de este documento "Marco Teórico" un prototipo no se centra en los requisitos estéticos, en lugar se centra en los requisitos funcionales que forman la base del sistema, es decir se debe poner mayor énfasis en los requisitos que son vitales para que el sistema opere adecuadamente, el resto de requisitos pueden ser descartables si no están relacionados directamente con el objetivo del desarrollo.

Se ha cuestionado a la muestra sobre el dispositivo de uso preferente para la visualización del contenido generado a través del prototipo, así como también las características que debería tener el mismo.

2. Teniendo en cuenta que el contenido puede ser visualizado únicamente desde un navegador web ¿Que dispositivo sería de su preferencia para la visualización del video en tiempo real? (El reproductor podría expandirse a pantalla completa).

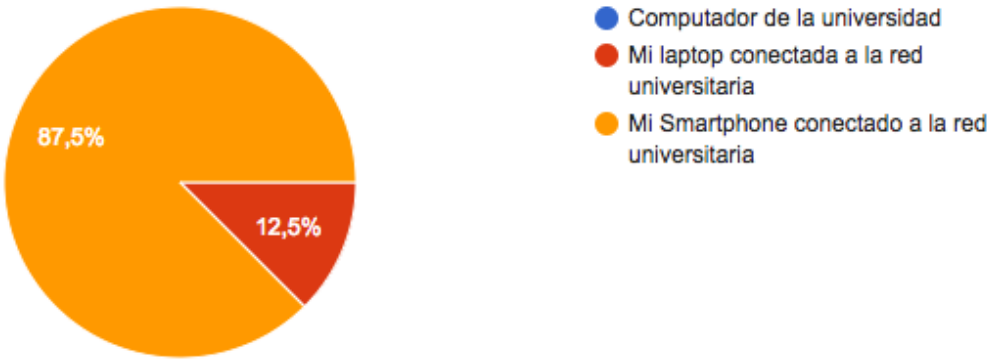


Figura 10. Requisitos estéticos *para* desarrollo del prototipo video streaming live en la FICA-UDLA.

Los resultados de esta parte de la encuesta nos dan una pauta sobre la plataforma de visualización de contenido a la cual deberíamos prestar prioridad, pero sin discriminar las otras opciones. Podemos interpretar el gráfico como un requerimiento multiplataforma sobre el prototipo a desarrollar.

A continuación, obtenemos los requerimientos o características adicionales que debería poseer el prototipo por medio de otro punto en la encuesta el cual fue presentado como:

3. ¿Qué características o funcionalidades cree que deberá tener este sistema?

Se ha diferenciado los requerimientos señalados por los encuestados como funcionales y no funcionales para tener una mejor perspectiva en el diseño e implementación del prototipo.

2.3 Requisitos funcionales

Son los requisitos a los cuales debemos prestar mayor importancia ya que de estos depende el comportamiento de nuestro sistema.

- El sistema debe ser fácilmente accesible para los estudiantes de la facultad universitaria.
- Retraso mínimo de tiempo entre el desarrollo de los eventos transmitidos y la visualización de estos por medio del sistema desarrollado.
- El sistema deberá transmitir el contenido sin interrupciones.
- El contenido emitido deberá guardarse automáticamente en el servidor.
- Se deberá poder usar tanto la cámara frontal como la posterior del dispositivo móvil para captar el video a ser emitido.

2.4 Requisitos no funcionales

Son requisitos de los cuales no depende el funcionamiento del prototipo, pero podrían ser incluidos en el desarrollo del mismo.

- Fácil interacción con el usuario.
- Bajo costo.
- La aplicación móvil debe estar provista de efectos o filtros de video.
- El video deberá poder ser compartido por medio de redes sociales.
- El video deberá ser editado previamente y de manera automática en el dispositivo móvil añadiendo un logo de la facultad o de la universidad.

2.5 Traducción de los requerimientos lenguaje técnico

Esta sección está destinada al análisis de los distintos formatos, estándares y protocolos que intervienen en el funcionamiento del prototipo enfocados al cumplimiento de los requerimientos más relevantes definidos en la sección anterior, consecuentemente se plasmará las dependencias de los mismos.

2.5.1 Sistema fácilmente accesible

El sistema debe superar las limitaciones de cableado, portabilidad y compatibilidad tanto para generar el contenido como para consumirlo. Todos estos componentes podrán comunicarse e interactuar por medio de la red cableada e inalámbrica que ofrece la universidad de las Américas.

2.5.2 Retraso mínimo y flujo ininterrumpido de la reproducción de contenido

Este requerimiento es dependiente de varios factores técnicos como son**:**

- Alto ancho de banda entre los componentes de la arquitectura.
- Baja latencia y jitter.
- Códec eficiente.

- Alta taza de procesamiento de contenido.
- Calidad de servicio.

2.6 Protocolos y formatos seleccionados

Para poder cumplir con los requerimientos del sistema y sus dependencias se realizó la elección de los protocolos y formatos en base a las funcionalidades, beneficios y facilidad de implementación que estos presentan para el prototipo.

2.6.1 RTMP

El protocolo de mensajes en tiempo real del cual Adobe Systems fue específicamente diseñado para transmitir contenido multimedia en tiempo real con una baja latencia, además permite el uso de streaming bajo demando y en directo atravez de una misma aplicación. Está basado en TCP para garantizar el envío de contenido de manera fluida diviendolo en fracmentos y negociando ventanas de transmisión de manera dinámica (Zingade, Joshi, Shendkar y Arora, 2016, pp. 2975-2977).

Este protocolo será usado para la comunicación entre la aplicación Android encargada de generar contenido y el servidor de multidifusión.

2.6.2 HTTP

Es un protocolo soportado por todos los navegadores web, los cuales se encuentran instalados tanto en Smartphones, SmatTV’s, laptops, etc. Debido al escalamiento y aceptabilidad de manera casi global en dispositivos que se encuentran en interacción con los usuarios, se ha elejido el protocolo HTTP como medio para visualizar el contenido multicast (Jestratjew y Kwiecien, 2013, pp. 271-276).

2.6.3 H.264

Es uno de los estándares de compresión mas avanzados reduce la cantidad de datos necesarios para transmitir el video usando una técnica denominada estimación del movimiento, está diseñado de tal manera que aunque los datos sin procesar lleguen a una relación cercana a 100:1 estos no puedan ser perceptibles por el usuario final (Bai, Song, Tao y Liu, 2013, pp. 500-503).

Este estandar será usado para codificar el contenido multimedia emitido por las cámaras y micrófono del Smatphone Android.

2.6.4 FLV

Es un contenedor altamente usado en plataformas como Youtube para transmitir video atravez de internet, FLV puede ser visualizado por la mayoría de dispositivos y sistemas operativos.

FLV será usado como contenedor para el codec H.264 debido a que es soportado en la gran mayoría de dispositivos (Vasanthi y Chidambaram, 2014, pp. 98-101)

2.6.5 HLS

HTTP live streaming es un protocolo de difusión de video en vivo que fue desarrollado por Apple que tiene como objetivo la difusión de contenido en tiempo real con tasas de bits adaptivas.

Debido a que los navegadores web de lo dispositivos móviles con Sistema Operativo Android en su mayoría no soportan FLV la mejor opción es usar el formato HLS que además es soportado por los navegadores web desarrollados por Apple (Yang, Choi y Kim, 2014, pp. 39-50).

2.7 Selección del software

Una vez definidas las necesidades del prototipo y los protocolos que más se ajustan para cubrir estas, debemos elegir el software tanto para el desarrollo de los componentes como para la puesta en marcha de la arquitectura a implementar por lo cual se tomará en consideración los parámetros definidos anteriormente.

2.7.1 Android Studio

Es un entorno de desarrollo integrado (IDE) oficial para aplicaciones Android, por lo cual brinda todas las herramientas y características para facilitar la codificación, entre las cuales tenemos:

- Es de uso gratuito.
- Interfaz gráfica y línea de comandos para el desarrollo.
- Facilita la reutilización de recursos y códigos.
- Importación sencilla de proyectos y códigos de ejemplo.
- Herramienta Pro Guard permite la reducción de código que se compila en la APK de manera que pueda consumir menos recursos en dispositivos de baja gama.
- Permite la ejecución de las aplicaciones desarrolladas tanto en el emulador integrado como en dispositivos móviles.

Android Studio será utilizado para el desarrollo del aplicativo Móvil Android encargado de generar el contenido de video y transmitirlo al servidor de multidifusión.

2.7.2 NGINX

Es un servidor libre y de código abierto que cumple, pero no se limita a las funciones de servidor web, proxy inverso web, proxy IMAP/POP3. NGINX provee alto rendimiento, estabilidad y bajo consumo de recursos, es utilizado en por grandes compañías de internet como lo son Netflix, SoundCloud, Facebook, y

demás debido a las prestaciones que este ofrece, a continuación, se mencionan características adicionales de este servidor (Liu, Wang, Wang, Gao y Zhu, 2015, pp. 991-995).

- Opciones de almacenamiento en caché.
- Puede proporcionar Balanceo de Carga.
- Soporte para IPv4 e IPv6 (Dual Stack).
- Módulos adicionales para ofrecer servicios como streaming.
- Soporta los formatos HLS y RTMP para streaming.
- Streaming adaptable sobre HLS.
- Multiplataforma.
- Altamente tolerante a fallos.
- Soporta hasta 10.000 conexiones simultáneas con baja utilización de recursos.
- IP basada en geo localización.

Este servidor ha sido seleccionado debido a su código abierto, gratuidad y sus prestaciones tanto de servicio web como de multidifusión de streaming capaz de soportar RTMP mismo protocolo en el que se recibirá el contenido multimedia. Adicionalmente podemos garantizar la entrega ininterrumpida de paquetes en los clientes que usen el protocolo HLS mediante streaming adaptable dependiente del ancho de banda disponible entre el cliente y el servidor NGINX (Taleb, Ksentini y Jantti, 2016, pp. 84-91).

2.7.3 JW Player

Es un reproductor de contenido multimedia en HTML5 como consecuencia este puede ser cargado tanto en computadores de escritorio, laptops, Smartphones e incluso en Smart TV. A continuación, se nombran las principales características de este reproductor Web (Liu, Wang, Huang, Wu, Wang, Jia y Chen, 2017, pp. 81-84).

- Fácil implementación.
- Trabaja sobre HTML5 proveyendo flexibilidad en el aspecto y origen del

contenido.

- Soporta HTTP, HTTPS, RTMP, RTSP, HLS.
- Posee una API JavaScript para la programación de funciones y controles.
- Soporta la maximización a pantalla completa.

La selección del reproductor de ha basado en sus características y ventajas como su fácil implementación, flexibilidad y soporte en los protocolos que forman parte de la arquitectura del protocolo “HLS y RTMP”.

2.7.4 Sistema Operativo UBUNTU

Es una distribución de Linux que es distribuida como software libre, y es actualizada cada 6 meses con la finalidad de realizar corrección de errores y bugs y además obtener los más recientes parches de seguridad, gracias a su código abierto y su descarga gratuita UBUNTU es desarrollado por la comunidad y para la comunidad. Existen versiones de esta distribución tanto de escritorio como de servidor, además la posibilidad de entorno gráfico o solamente consola (Zhou, Xu y Wang, 2017, p. 13).

El sistema operativo UBUNTU fue seleccionado como base para el servidor web y streaming considerando sus características como su gratuidad, compatibilidad con NGINX y la gran cantidad de documentación disponible en internet.

3. CAPÍTULO III. DISEÑO DEL PROTOTIPO

Esta sección tiene como objetivo el diseño de la solución a partir de los recursos de software seleccionados en el capítulo anterior y la interacción entre cada una de ellos, adicionalmente se definirá los componentes físicos que harán posible el funcionamiento del prototipo plasmados en un diseño físico.

3.1 Diseño lógico del prototipo

Este diseño define la forma y acciones a seguir para el cumplimiento de los requisitos definidos en la sección del análisis, así como también la interacción que tendrá el usuario con el sistema a través de casos de uso e interfaces.
Con la finalidad de facilitar el entendimiento y diseño de la solución se han definido 4 módulos que incluyen distintos componentes y están encargados de funciones específicas.

3.1.1 Módulo generador de contenido

El módulo generador de contenido está compuesto únicamente por el aplicativo Android, mismo que está encargado de captar el audio y video del dispositivo móvil para posteriormente encapsularlo o codificarlo en formato FLV y transmitirlo hacia el siguiente módulo mediante el protocolo RTP.

3.1.2 Módulo de procesamiento de video

En este módulo se recibirán las transmisiones de video enviadas por el módulo generador de contenido a través del protocolo RTP para posteriormente ser codificadas en varios formatos de calidad con la finalidad de brindar una calidad de servicio a nivel de aplicación, mientras que una copia de la transmisión original será almacenada localmente a manera de respaldo para cumplir con los requerimientos provistos en las encuestas.

Adicionalmente este módulo será el encargado de verificar el ancho de banda disponible de las conexiones con el módulo web, aprovechando este valioso recurso se deben definir al menos tres calidades de video adicionales mismas que serán utilizadas de manera variante y dependiente del ancho de banda disponible con la finalidad de brindar un servicio eficiente e ininterrumpido.

El aplicativo NGINX será el encargado de gestionar las características anteriormente señalas mediante la adición de los módulos de RTP y HLS mismos que han sido diseñados para la transmisión de video streaming.

3.1.4 Módulo Web

Este módulo estará funcionando sobre NGINX y será el encargado de atender las peticiones generadas por los dispositivos clientes y proveyendo una página web basada en HTML5 para la visualización de contenido de video por medio del reproductor web JWplayer.

Este módulo debe atender las peticiones web generadas por los clientes y notificar al módulo de procesamiento de video el ancho de banda disponible entre las conexiones generadas, de esta manera cada conexión existente tendrá su calidad de video diferenciada mediante los recursos de red que esta disponga.

3.1.5 Módulo Cliente

Este módulo se encuentra compuesto por la gran variedad de dispositivos móviles, laptops, tablets, etc. y los navegadores web que pueden poseer. Su función consiste en realizar peticiones al servidor web y consumir los recursos que este brinda es decir transmisiones de video en vivo por medio de HTML.

Habiendo explicado la funcionalidad de cada módulo y los medios y servicios que estos proveen y consumen. A continuación, se presenta el gráfico de la arquitectura lógica que presenta el prototipo a desarrollar. A continuación la figura 11 representa la arquitectura base a implementar.

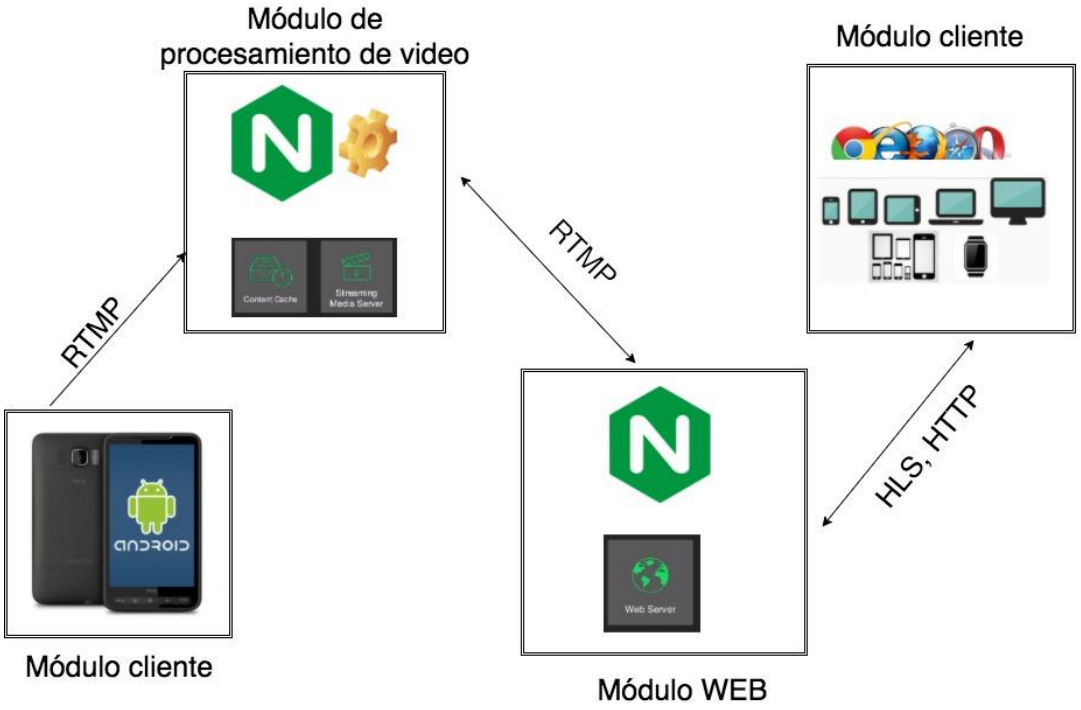


Figura 11. Arquitectura lógica del diseño a seguir.

3.2 Casos de uso

Un caso de uso detalla los pasos a seguir para cumplir una funcionalidad, la cual se realiza conjuntamente con el o los distintos elementos que intercambian información con el sistema mismo que son denominados como actores. Los casos de uso manifiestan el que debe hacerse mas no señalan la manera para hacerlo.

Existen dos actores claramente diferenciables los cuales interactúan con el sistema de video streaming en vivo, mismo que son denominados como usuario generador de contenido y usuario receptor de contenido, tal como indican los nombres sus funciones son generar el contenido de video o limitarse a la visualización del mismo.

A continuación, se representa el único escenario que puede existir en el que se detalla la interacción de los usuarios con el sistema de video streaming en vivo.

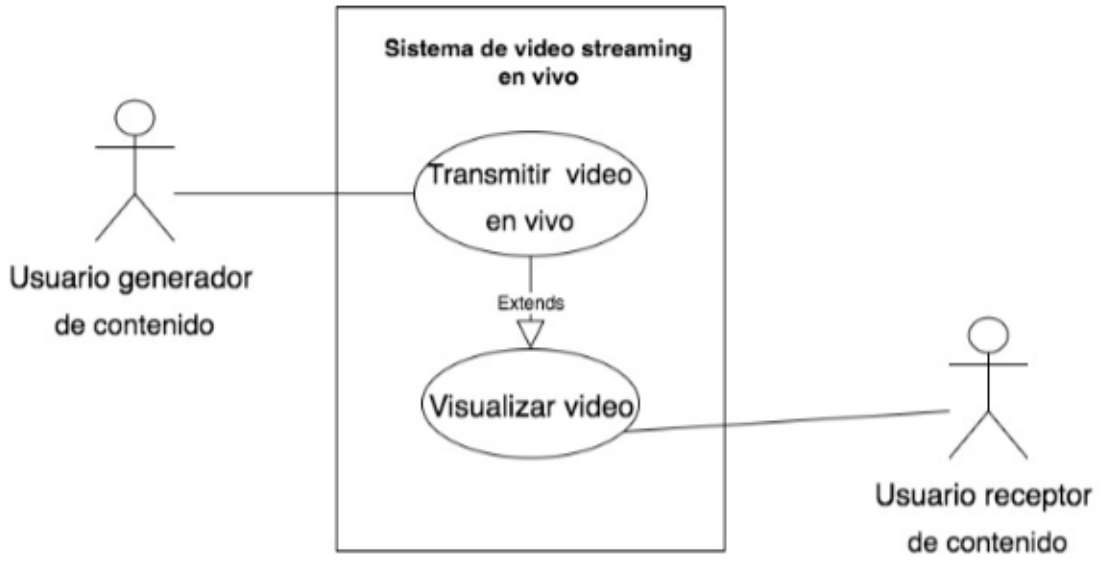


Figura 12. Interacción de los usuarios con el sistema de video streaming en vivo.

- Transmitir video. - En usuario abre la aplicación Android, elije la cámara que desea usar y oprime el botón transmitir, en cuanto la conexión entre el dispositivo Android y el servidor sea establecida el video será transmitido.
- Visualizar video streaming. - El usuario abre cualquier navegador web disponible e ingresa la URL o IP del servidor de streaming en la barra de direcciones, como consecuencia el contenido de video en tiempo real será desplegado para poder ser visualizado.

3.3 Diseño de interfaces del sistema

Para facilitar el diseño y despliegue del prototipo se diseñarán únicamente dos interfaces para la interacción del usuario con el sistema.

3.3.1 Interfaz Android generadora de contenido

Esta interfaz se encuentra contemplada en el aplicativo Android y debe estar diseñada de tal manera que pueda cumplir con las características definidas en el capítulo posterior, estos requerimientos se pueden resumir en:

- Rápida ejecución. - El usuario requiere acceder al aplicativo y que esté disponible para realizar la transmisión tan pronto como sea posible.
- Uso de cámaras. - Se requiere que el aplicativo pueda realizar el intercambio de cámaras del dispositivo cuando el usuario lo requiera.
- Calidad de transmisión. - Se puede transmitir este requerimiento al diseño de la interfaz al seguir las recomendaciones generales para grabar video, las cuales concluyen que los videos captados desde un dispositivo móvil deben tomarse de forma horizontal. Adicionalmente se incluirá la opción de usar el flash de la cámara si esta la posee.

En la figura 13 se presenta el diseño de la interfaz Android, con los componentes que facilitarán el cumplimiento de los requerimientos.

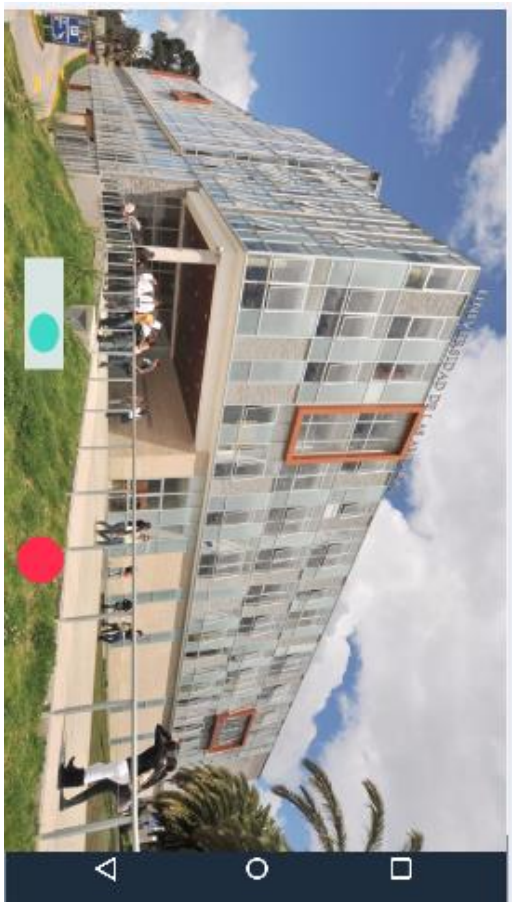


Figura 13. Ejemplo de la interfaz desde un dispositivo móvil (celular).

Esta interfaz está compuesta por los siguientes componentes:

- Pantalla completa con captura de cámara. - Está diseñada de manera

que despliegue la cámara posterior del dispositivo Android cubriendo toda la pantalla con excepción de la barra de navegación.

- Switch cámara. - Este botón será el encargado de alternas las cámaras frontal y posterior.
- Botón grabar. - Es botón se presenta de color gris cuando no se ha presionado y cambiará a rojo en caso contrario, es responsable de establecer la comunicación e iniciar la transmisión de video hacia el módulo procesador de contenido.

Como se puede apreciar en el diseño los botones están ubicados de manera que induzcan al usuario a colocar el dispositivo de manera horizontal para grabar el contenido, adicionalmente se han establecido los siguientes mensajes para guiar al usuario:

- Coloca tu dispositivo de manera horizontal. - Este mensaje será desplegado en cuando el aplicativo Android sea abierto.
- Error de conexión. - Se presenta este mensaje cuando existe un problema de conexión.
- Transmisión Iniciada. - Tras oprimir el botón de grabar y haberse establecido la conexión se desplegará este mensaje indicando que el contenido que se está capturando ya puede ser accedido.
- Transmisión finalizada. - Este mensaje será desplegado al usuario cuando exista una interrupción voluntaria o problema en la transmisión del contenido de video.

3.3.2 Interfaz Web de visualización de contenido

La interfaz Web se encuentra configurada en el módulo web pero el consumo de este servicio se lo hace por medio de un navegador WEB mismo que se encuentra comúnmente instalado nativamente en dispositivos celulares actuales, Smart TV, etc.

Si bien es cierto el acceso cotidiano a una página web en particular es realizado a partir del ingreso del nombre del sitio a visitar en la barra de búsqueda del navegador, al encontrarnos en el desarrollo de un prototipo no se dispone de un servicio DNS, mismo que es usado para asociar direcciones IP a nombres para facilitar la administración y acceso a sitios web y dispositivos en general. Consecuentemente a lo mencionado haremos uso únicamente de la dirección IP del servidor WEB para poder ingresar a la página de visualización del video streaming emitido en vivo.

Adicionalmente se tiene que tener a consideración que el desarrollo del prototipo no tiene un alcance en cuanto a seguridad se refiere, por tal motivo la carga del aplicativo web se realizará bajo http, mismo que se define como protocolo de transferencia de hipertexto y no provee ningún mecanismo de seguridad a diferencia de https.

Referentemente a los requisitos de esta interfaz podemos definir los siguientes:

- Rápido despliegue. - Para cumplir con este requerimiento dependemos enteramente de las capacidades asignadas a nuestro servidor por medio de la UNIVERSIDAD DE LAS AMÉRICAS.
- Visualización pantalla completa. - Los usuarios requieren que la visualización del contenido pueda realizarse con la amplitud permitida por el navegador WEB en uso.

En la siguiente figura se puede apreciar el diseño que tendrá la página web a fin de cumplir con los requerimientos definidos por los usuarios.

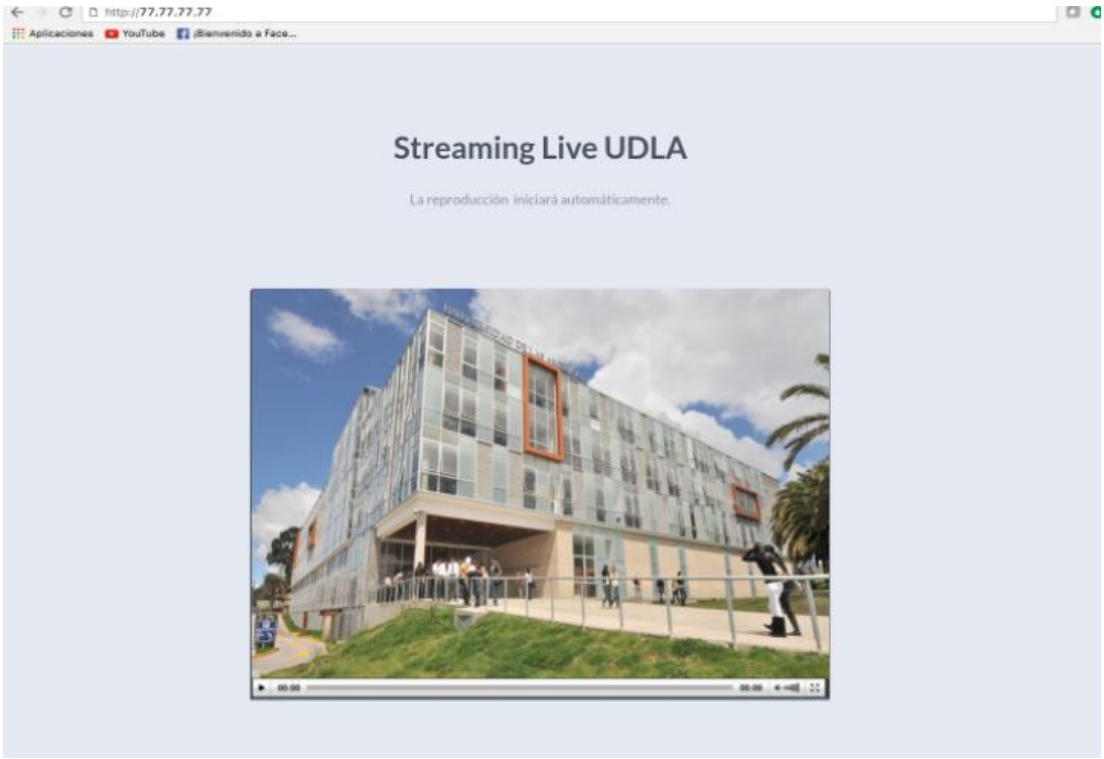


Figura 14. Interfaz web de visualización de video streaming en vivo

Como se puede apreciar en la figura el componente principal de esta interfaz es el reproductor WEB basado en HTML5, mismo que tiene las siguientes características:

- Reproducción automática del video streaming en vivo.
- Botón de ampliación del reproductor a pantalla completa.

4. CAPÍTULO IV. IMPLEMENTACIÓN DEL PROTOTIPO Y REALIZACIÓN DE PRUEBAS

Esta etapa del proyecto consiste en la descripción paso a paso de los procesos más relevantes en el desarrollo de cada uno de los módulos que conforman la solución, debido a la dependencia que existe entre los componentes del prototipo y la facilidad para la realización de pruebas funcionales se realizará un desarrollo secuencialmente siguiendo este orden:

- Módulo de procesamiento de datos.
- Módulo web.
- Módulo generador de contenidos.

4.1 Desarrollo del módulo generador de contenidos

Este módulo debe ser el primero en ser implementado debido a la importancia que presenta para el prototipo al ser responsable de decodificar y codificar el contenido recibido a varios formatos y calidades con el fin de mejorar la calidad de la experiencia del usuario. Existen varias dependencias mencionadas en el capítulo 2 “Análisis de Requerimientos, elección de protocolos y software”, las cuales debemos tener a consideración.

4.1.1 Instalación y configuración del Sistema Operativo Base

Como se mencionó en capítulos anteriores el sistema operativo seleccionado es Ubuntu Server TLS, debido a la estabilidad y eficiencia que ha sido previamente comprobada y documentada en distintos foros de internet.

Este módulo del sistema está diseñado y limitado para funcionar sobre el data center de la FICA en un ambiente compartido, es decir que es necesario el uso de un virtualizador que aloje el servidor y nos provea escalabilidad, en este caso usaremos VMware por su flexibilidad y la fácil migración a otras plataformas como hyper-v y virtual Box.

4.1.1.1 Creación de la máquina virtual

Podemos obtener una imagen del sistema a virtualizar simplemente ingresando a la página oficial de Ubuntu e indagando entre la sección de descargas hasta encontrar el sistema operativo a descargar, no sin antes tener a consideración que esta imagen debe ser apta para ser montada sobre equipos físicos Intel ya que se hará uso de un chasis Cisco el cual está provisto con procesadores Intel. A manera de sugerencia Ubuntu provee un listado de capacidades mínimas para este sistema en las que se incluyen:

- 2 cores para Procesador.
- 2 GB de Memoria RAM.
- 25 GB de Espacio de libre en disco.

El proceso de creación de la máquina virtual consta de únicamente de la carga del sistema operativo y asignación de recursos a la máquina virtual, en nuestro caso usaremos recursos por encima de los sugeridos por Ubuntu debido a la cantidad de procesamiento que debe soportar nuestro servidor, el almacenamiento en memoria cache, y el disco a utilizar para el almacenamiento de videos.

Tabla 1.

Recursos asignados a servidor virtual de streaming

RECURSOS ASIGNADOS AL SERVIDOR DE STREAMING		
Procesador	Memoria RAM	Disco Duro
4 Core	4 GB	50 GB

Dado que la información sobre las capacidades necesarias en el servidor para la codificación y decodificación en NGINX no se encuentran disponibles. A continuación, se dispone del justificativo para la elección de estas (García, Galán y Izquierdo, 2017, pp. 37-51).

- Procesador. - Debido a la excesiva carga que presentará nuestro servidor y los requerimientos de rápido procesamiento se ha elegido una base de 4 cores o núcleos cuya función será procesar los distintas decodificaciones y codificaciones. Resumidamente un mayor número de cores, refleja un mejor procesamiento.
- Memoria RAM. - El mínimo de memoria RAM sugerido para servidores de streaming en general es de 8 GB.
- Disco Duro. - Este recurso será utilizado principalmente para alojar es sistema operativo el cual ocupa aproximadamente 8 GB y para al almacenamiento de los streaming generados por el aplicativo Android.

4.1.1.2 Configuración del Sistema Operativo

En cuanto haya finalizado la creación de la máquina virtual el paso a seguir es la configuración y personalización del sistema operativo, al iniciar al sistema operativo se pide ingresar la clave para el usuario root (usuario del sistema con todos los privilegios sobre ficheros, servicios, etc.) y adicionalmente se puede agregar un usuario que posteriormente puede ser configurado para tener todos los mismos privilegios que el usuario root, lo cual es visto como una buena práctica para no perder administración total sobre el sistema en caso de pérdida o eliminación de credenciales de usuarios.

Se ha agregado el usuario NGINX el cual ha sido incluido en el listado de usuarios administrativos mediante la edición del archivo sudousers, tal como se indica en la figura 15.

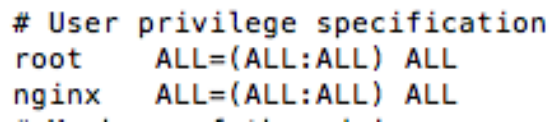


Figura 15. Usuarios con privilegios del sistema.

Es deseable brindar una conexión remota por consola hacia el servidor ya que facilita la administración de este desde lugares alejados al lugar de implementación, se ha elegido el servicio de conexión remota segura SSH para cumplir este propósito, el cual no se encuentra instalado por defecto en esta distribución de Ubuntu, consecuentemente este servicio debe ser instalado por medio del comando “apt-get install openssh-server”, al finalizar la instalación podemos iniciar y verificar el estado del servicio (ver figura 16).

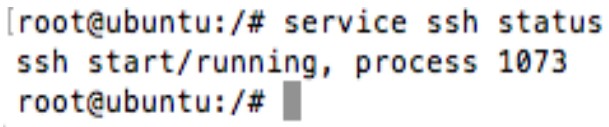


Figura 16. Estado del servicio SSH.

4.1.2 Instalación y configuración de NGINX

Ya que NGINX está basado en código abierto este puede ser descargado de variedad de sitios, pero siempre será preferible hacerlo desde su página oficial. Debemos recordar que este software es la base de nuestro sistema por lo cual es de vital importancia la instalación de todas sus dependencias y librerías para un correcto funcionamiento del servicio.

4.1.2.1 Solución de dependencias y compilación de Software.

Antes de iniciar la instalación de NGINX debemos descargar una lista de compiladores necesarios, los cuales no están incluidos en la distribución de Ubuntu de la cual se está haciendo uso, las librerías de compilación a descargar e instalar son:

- built-essential
- libpcre3
- libpcre3-dev
- libssl-dev

Una vez instaladas las dependencias, podemos continuar con la descarga y compilación del software, como ya se mencionó este será obtenido por medio de la página oficial de NGINX en su versión 1.7.5. Se debe tener presente que NGINX en su descarga tradicional nos ofrece un servicio de páginas web del cual estará encargado Apache y no un servicio de procesamiento de streaming el cual es deseado, como consecuencia debemos descargar adicionalmente el módulo RTMP el cual estará encargado de receptar el streaming de video en vivo y procesarlo según las especificaciones requeridas.

Habiendo realizado las descargas necesarias podemos descomprimir y compilar NGINX adicionando el módulo RTMP, para ello nos ubicamos en el directorio descomprimido del aplicativo y emitimos los siguientes comandos de manera secuencial:

- ./configure --add-module=../nginx-rtmp-module-master
- make
- make install

Si la compilación e instalación no ha presentado errores NGINX se encontrará en el directorio “/usr/local/nginx” donde se ubican los archivos de configuración que serán usados posteriormente en el desarrollo de este documento.

4.1.2.2 Configuración de NGINX

Antes de iniciar con la configuración del software hay que tener en cuenta varias consideraciones basadas en los requerimientos de usuario y eficiencia de codificación/ decodificación, las cuales son detalladas a continuación:

1. Una copia del streaming original será almacenada localmente.
2. El video visualizado en móviles debe ser escalado a una menor resolución con la finalidad de tener un mejor uso del ancho de banda disponible.

La configuración de este software consta únicamente de la edición del archivo nginx.conf el cual nos facilita de gran manera la personalización de procesamiento del streaming, en esta sección se editará tanto la codificación HLS para móviles y dispositivos Apple como la RTMP para laptops y computadores de escritorio.

En la figura 17 se muestra la configuración de los aplicativos live, mobile y vod.

```
application live {
        allow play all;
        live on;
        record all;
        record_path /video_recordings;
        record_unique on;
        hls on;
        hls_nested on;
        hls_path /HLS/live;
        hls_fragment 1s;
        hls_playlist_length 2s;
        hls_type live;
        hls_continuous off;
        hls_cleanup on;
        #reduccion de resolucion del streaming en ingreso y publicacion en la aplicacion moviles
        exec_push /home/administrador/bin/ffmpeg -i rtmp://localhost/live/$name -threads 1 -c:v libx264 -profile:v baseline
         -b:v 350K -s 640x360 -f flv -c:a aac -ac 1 -strict -2 -b:a 56k rtmp://localhost/mobile/$name 2>>/var/logs/$name.log;
        }
#aplicacion para moviles "menor resolucion"
application mobile {
        allow play all;
        live on;
        hls on;
        hls_nested on;
        hls_path /HLS/mobile;
        hls_fragment 1s;
        hls_playlist_length 2s;
        hls_type live;
        hls_continuous off;
        hls_cleanup on;
        }
#aplicacion para videos bajo demanda, permite la lectura de videos almacenados en el directorio video_recordings
application vod {
        play /video_recordings;
        }
```

Figura 17. Configuración de NGINX

Parámetros de configuración comunes de los aplicativos nginx

- allow play all. - Permisos de reproducción a todas las conexiones.
- live on. - Habilita la propagación broadcast de los streaming.
- hls_nested. - Un subdirectorio es creado por cada streaming emitido.
- hls_path. - Define el directorio destino para los archivos temporales de streaming.
- hls_fragment. - Limita el tamaño en segundos de los fragmentos con extensión ts.
- hls_playlist_length. - Define el tiempo de almacenamiento de los archivos ts.
- hls_type. - Asegura que la duración del playlist sea suficiente para la emisión del streaming en el modo seleccionado.
- hls_continous. - Permite que la reproducción del streaming se reanude desde donde se pauso o corto.
- hls_cleanup. - Remueve de la memoria cache los fragmentos ts.

4.1.2.3 Aplicativo live

Recibe el streaming original de video en vivo emitido por el aplicativo celular Android bajo RTMP, a continuación, éste será almacenado localmente en el directorio video_recordings con un nombre único para cada emisión. Esta trasmisión estará disponible en formado m3u8 para poder ser visualizado en dispositivos Apple y en formato flv para laptops y computadores de escritorio.

NGINX nos ofrece la posibilidad de ejecutar aplicativos o comandos externos, aprovechando esta característica se hará uso del software ffmpeg el cual ha sido previamente instalado, ffmpeg será utilizado para reducir la resolución de los videos que serán consumidos por dispositivos móviles, tras realizada la conversión será publicada en el aplicativo mobile.

4.1.2.4 Aplicativo mobile

Recibe el streaming de baja resolución y lo codifica con la extensión m3u8 para poder ser visualizado usando el protocolo HLS.

4.1.2.5 Aplicativo vod

Sus siglas derivan de las palabras en inglés Video Off Demand, permite la visualización del contenido de video almacenado en el directorio /video_recordings.

Se puede verificar que las transmisiones se estén almacenando localmente, así como también que la conversión de resolución de video en tiempo real se esté ejecutando adecuadamente.

4.1.3 Prueba modular de codificación de video

Para generar el contenido de prueba en formado RTMP, se utilizó el aplicativo adobe flash media live encoder, la siguiente figura demuestra que efectivamente el contenido es receptado y almacenado localmente en el servidor.

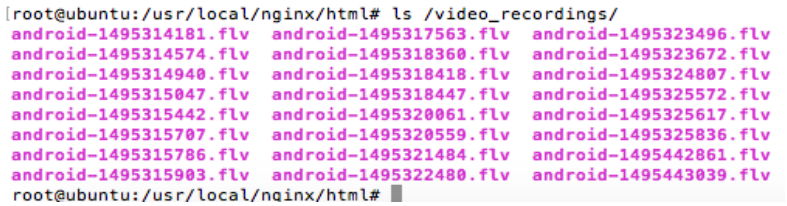


Figura 18. Prueba modular de codificación de video.

En la Figura 18 podemos observar que la codificación se esté realizando adecuadamente. Adicionalmente, se han re direccionado los logs de procesamiento hacia la ubicación /var/logs/android.log, este archivo puede ser visualizado en tiempo de ejecución mediante el comando "tail -20f /var/logs/android.log" tal como se aprecia en la figura 19.

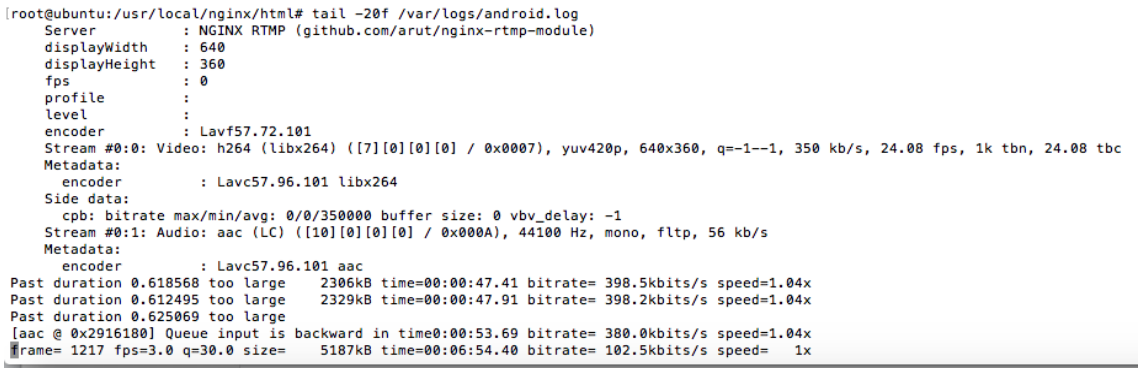


Figura 19. Direccionamiento de los logs a ubicación /var/logs/android.log

4.2 Desarrollo del módulo WEB

Una vez que el módulo generador de contenido ha sido implementado y se ha verificado su funcionamiento se puede proceder con el desarrollo del aplicativo para la visualización de los videos en tiempo real, como se ha mencionado con anterioridad NGINX es principalmente un servidor WEB por lo cual su uso es conveniente para evitar problemas de compatibilidad y retraso de procesamiento provocado por las transacciones y el cacheo de información que pueden generar el uso de más aplicativos.

Ya que el software NGINX se encuentra instalado se requieren las tareas de configuración del archivo nginx.conf, solución de dependencias y codificación HTML5 para que este módulo funcione de acuerdo a los requerimientos.

4.2.1 Resolución de dependencias de página HTML

En esta etapa del desarrollo se obtendrán, instalaran y configurar las varias dependencias funcionales y no funcionales que están relacionadas con el aplicativo WEB (Zhou et al., 2017).

- Reproductor HTML5. - Podemos minimizar la utilización de recursos usando el navegador por defecto que incluye HTML5 mediante la etiqueta video, lamentablemente este reproductor no soporta el protocolo rtmp, así que será utilizado para móviles y dispositivos Apple.
- Reproductor JW Player. - Este reproductor estará hosteado (localizado) en el servidor virtualizado, por lo cual se debe obtener los archivos básicos de funcionamiento del reproductor. La descarga puede hacerse tras realizado el registro en la página oficial del reproductor donde se proveen todas las versiones y credenciales autorizadas de uso. JW Player será usado únicamente para la visualización de contenido en laptops y computadores de escritorio.
- Estilo metro. - Un estilo es de ayuda en el diseño ya que define un formato o estructura de presentación para la página WEB, de esta manera se evita la definición de tipo de letra, tamaño, etc. En este caso se descargará el estilo Metro obtenido desde su página oficial.
- Jquery. - En su definición básica es un conjunto de librerías Javascript, al incluirlo en la codificación HTML facilitará la manipulación de los reproductores de video, identificación de plataformas y definición de variables.

4.2.2 Codificación HTML

Una vez solucionadas las dependencias del aplicativo WEB se procede con la codificación del mismo haciendo uso de HTML5, JavaScript y los recursos mencionados en la sección anterior.

En las figuras 20 y 21 se da a conocer tanto el código en lenguaje de etiquetas empleado como el uso de las funciones JavaScript.

```
<html>
    <head>
        <title>Streaming Udla</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="js/jquery.js" ></script>
        <script src="jwplayer/jwplayer.js"></script>
        <link href="metro/css/metro.min.css" rel="stylesheet" />
        <link href="metro/css/metro-icons.min.css" rel="stylesheet" />
        <link href="metro/css/metro-responsive.min.css" rel="stylesheet" />
        <link href="metro/css/metro-schemes.min.css" rel="stylesheet" />
    </head>
    <body style="background: url(img/background.jpg) center fixed; background-size: cover;">
        <div class="container">
            <div class="grid">
                <div class="row">
                    <div class="cell">
                        <img src="img/logo.png" class="place-right" height="80" style="height: 80px; width: auto;" />
                    </div>
                </div>
                <div class="row rtmp">
                    <div class="cell">
                        <div id="player2" style="width:100%;"></div>
                    </div>
                </div>
                <div class="row hls">
                    <div class="cell">
                        <div data-role="video" data-full-screen-mode="default">
                            <video>
                                <source id="ruta_video" src="http://192.168.1.37/live/android/index.m3u8" type="application/x-mpegURL" />
                            </video>
                        </div>
                    </div>
                </div>
            </div>
        </div>
```

Figura 20. Codificación de etiquetas HTML5 aplicativo WEB

```
<script>

    jwplayer.key = "IzEqVjRNGbvR6o5C9Fa0V+d5RKsU6WMks6OoUQ==";
    var ip = '192.168.1.37';
    var is_chrome = navigator.userAgent.indexOf('Chrome') > -1;
    var is_explorer = navigator.userAgent.indexOf('MSIE') > -1;
    var is_firefox = navigator.userAgent.indexOf('Firefox') > -1;
    var is_safari = navigator.userAgent.indexOf("Safari") > -1;
    var is_opera = navigator.userAgent.toLowerCase().indexOf("op") > -1;
    if ((is_chrome) && (is_safari)) {
        is_safari = false;
    }
    if ((is_chrome) && (is_opera)) {
        is_chrome = false;
    }
    var mobile = /Android/i.test(navigator.userAgent);

    $(document).ready(function () {
        if (mobile || is_safari) {
            if(mobile){
                $("#ruta_video").attr("src","http://"+ip+"/mobile/android/index.m3u8");
            }
            $(".rtmp").hide();
        } else {
            $(".hls").hide();
        }
    });

    jwplayer('player2').setup({
        file: 'rtmp://' + ip + '/live/android',
        image: '/hls/paris.jpg',
        title: 'Mrwhitep',
        width: '100%',
        height: '500',
        aspectratio: '10:9',
        autostart: 'true',
        fallback: 'true',
        primary: 'flash'
    });

</script>
```

Figura 21. Uso de funciones jquery en aplicativo WEB.

Debido a que se está haciendo uso de dos reproductores simultáneamente en un el mismo aplicativo web, sería erróneo presentarlos al usuario final de esta manera, consecuentemente mediante el uso de jquery se realiza la identificación de la plataforma desde la cual se está accediendo al aplicativo web, es decir al

hacer la evaluación de dicha variable se ocultará el reproductor que no será utilizado.

4.2.3 Configuración NGINX WEB

Como se mencionó con anterioridad la configuración y personalización de NGINX se centra en el archivo nginx.conf por lo cual será editado nuevamente a fin de añadir las características requeridas.

En esta situación es conveniente tener la lectura del archivo con extensión m3u8 en cuanto este ha sido modificado ya que contiene el orden de los fragmentos disponibles para ser reproducidos, adicionalmente el contenido no debe ser almacenado en cache ya que causaría retardo y una reproducción distinta a la definida como tiempo real.

Parámetros comunes de configuración web

- if_modified_since. - Indica la fecha y hora de la última modificación de la cabecera http.
- add_header. - Define una directiva para la cabecera http, se puede definir un nombre, fecha de modificación, almacenamiento en cache, etc.

La figura 22 a continuación muestra la configuración del aplicativo, principalmente se pretende eliminar el cacheo en los directorios temporales del protocolo HLS (Nedelcu, C., 2017, p. 70).

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

server {
        listen 80;
        server_name 192.168.1.37;
        #especificacion de formato, ubicacion y parametros de almacenamiento en cache para el aplicativo live en http
        location /live {
                types {
                        application/vnd.apple.mpegurl m3u8;
                        }
        alias /HLS/live;
        if_modified_since off;
        add_header Last-Modified "";
        add_header Cache-Control "no-cache";
        etag off;
        }
        #especificacion de formato, ubicacion y parametros de almacenamiento en cache para el aplicativo movile en http
        location /mobile {
                types {
                        application/vnd.apple.mpegurl m3u8;
                        }
        alias /HLS/mobile;
        if_modified_since off;
        add_header Last-Modified "";
        add_header Cache-Control "no-cache";
        etag off;
        }
        #permite el uso directo de la pagina index
        location / {
                root   html;
                index  index.html;
                }
        }
}
```

Figura 22. Configuración NGINX web

Como se puede apreciar en la figura previa la configuración web está diseñada de manera modular para cada aplicativo a ser utilizado, facilitando el entendimiento y flexibilidad de la codificación.

Al finalizar la configuración de NGINX y el desarrollo del aplicativo web el módulo estará listo para la visualización de contenido.

4.2.4 Prueba modular del aplicativo web

Se puede verificar el funcionamiento del aplicativo web al realizar una emisión y visualizarla en los navegadores provistos en distintos dispositivos, como muestra de la versatilidad del prototipo se han tomado distintas plataformas base para esta prueba modular (ver figura 23).

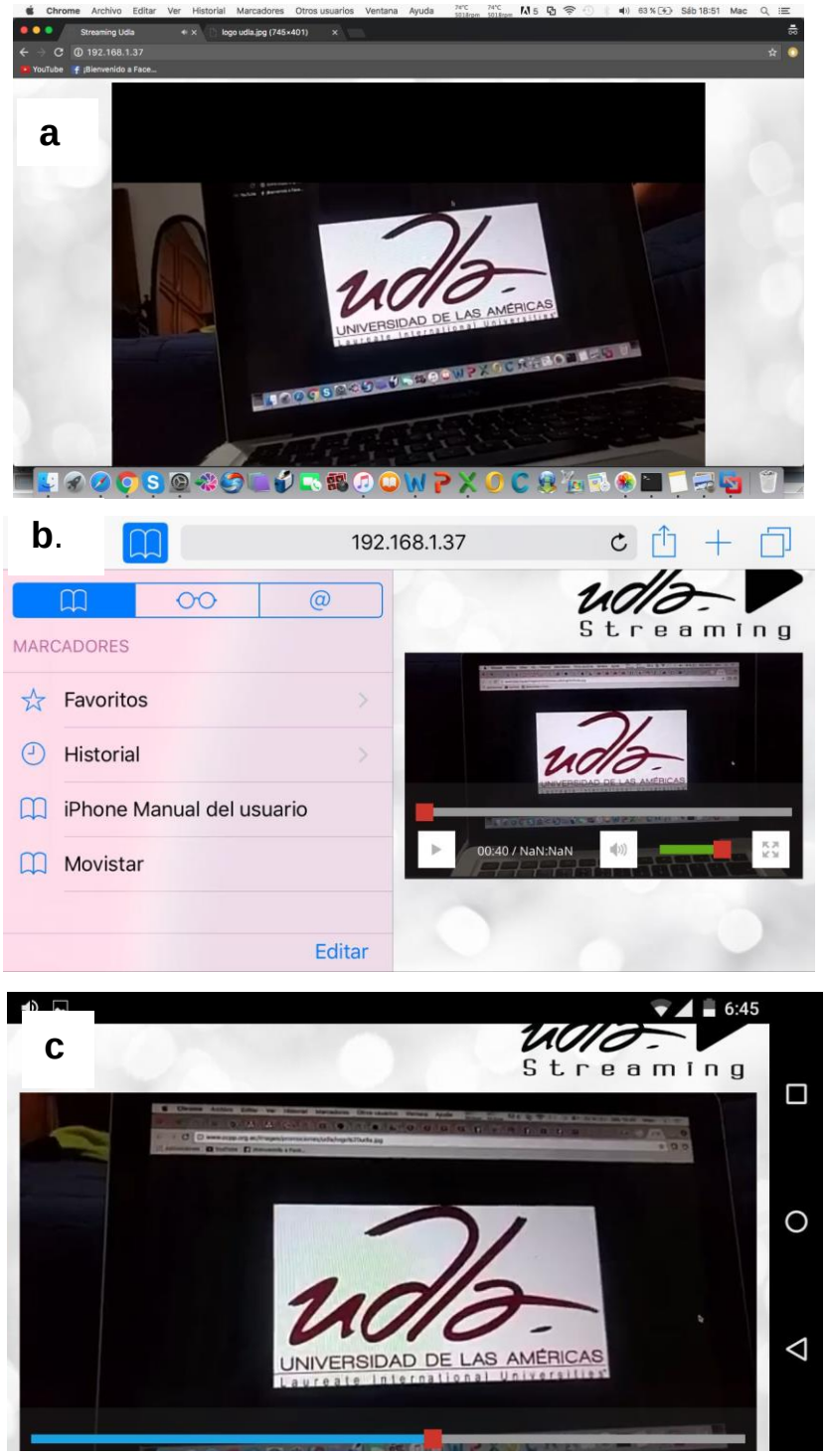


Figura 23. Prueba modular del aplicativo web

a. Prueba macbook pro navegador Chrome.
b. Prueba IOS iPhone 7 navegador safari.
c. Prueba Android Nexus 5x navegador Chrome.

Tal como se puede apreciar el video en vivo puede ser visualizado desde cualquier dispositivo, de esta manera se ofrece al usuario un servicio portable y eficiente con soporte en todas las plataformas.

4.3 Desarrollo módulo generador de contenidos

En esta sección del documento se desarrollará el aplicativo Android, ya que este aplicativo debe captar el contenido provisto por la cámara y micrófono del dispositivo móvil para luego transmitirlo bajo el protocolo RTMP, el primer paso consta de la búsqueda de una librería capaz de cumplir con estos requerimientos, lamentablemente la gran mayoría de estas se encuentran licenciadas bajo versiones de pago. Gracias a la plataforma en línea GitHub la cual provee una plataforma para subir proyectos de código abierto, se ha localizado el aplicativo Android Yasea.

4.3.1 Aplicativo Android Yasea

Este aplicativo se encuentra alojado en la plataforma GitHub bajo una licencia de código abierto la cual permite a la comunidad en general el hacer uso, modificación o distribución del mismo mientras esta no sea con fines comerciales. Yasea ha sido desarrollado por Leo Ma miembro activo de la comunidad GitHub, el cual describe al aplicativo como un cliente Android de streaming el cual codifica los datos YUM y PCM obtenidos la cámara y micrófono hacia H.264/ACC, los encapsula en formato FLV y realiza la transmisión sobre RTMP. Adicionalmente se describen las siguientes características del aplicativo (Gitfub, Inc., 2017):

- Android mini API 16.
- H.264/AAC hard encoding.
- H.264 soft encoding.
- Flujo de RTMP con controlador de estado de devolución de llamada.
- Orientación pantalla dinámica.
- Intercambio manual de cámaras frontal y posterior.
- Grabación local del streaming emitido en formato MP4

- Filtros de video.
- Cancelación de eco acústico y control automático de ganancia.

Como se puede apreciar Yasea provee la mayoría de características requeridas en el prototipo, por lo cual este será modificado a fin de realizar las codificaciones adicionales y personalización del aplicativo para la universidad de las Américas.

4.3.2 Edición de Yasea

Después de realizada la descarga del aplicativo Yasea se puede constatar que este se encuentra provisto de las librerías necesarias para la manipulación de los recursos de cámaras, pantalla y recursos de streaming. Se debe tener en consideración que las características del aplicativo señaladas en la sección previa no serán implementadas en su totalidad ya que están fuera de las necesidades transmitidas por los usuarios en las encuestas realizadas, consecuentemente las características a implementar son las siguientes:

- H.264 soft encoding.
- Pantalla fijada horizontalmente.
- Intercambio manual de cámaras frontal y posterior.

Inicialmente se deben definir los permisos que debe tener el aplicativo sobre el dispositivo instalado, así como también la orientación de la pantalla, estos atributos están definidos sobre el manifest AndroidManifest.xml detallado en la figura 24.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="ec.com.udla.tesis">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
    <uses-feature android:glEsVersion="0x00020000" android:required="true" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Udla Stream"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:screenOrientation="landscape"
            android:name="MainActivity"
            android:configChanges="orientation|screenSize">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Figura 24. AndroidManifest de aplicación streaming.

Toda la codificación se encuentra alojada en la clase principal MainActivity el cual hace uso extensible de las clases definidas en las librerías incorporadas, a continuación, se detalla la función de los principales métodos y atributos de los botones usados (Gitfub, Inc., 2017):

- onCreate. - Inicialización de valores en variables y métodos tales como resolución, elección de cámara, eliminación de la barra de acción, etc.
- onPause. - Detiene la transmisión de contenido en cuanto la ejecución normal de la aplicación es interrumpida por una llamada u otro aplicativo.
- onClick sobre el botón publicar. - La función principal de este botón consiste en iniciar la captura y transmisión del contenido hacia el servidor de codificación/decodificación mediante un identificador String que dispone la información de dirección ip del servidor, aplicación NGINX a utilizar y nombre identificador de la transmisión. Adicionalmente, el botón culmina toda transmisión en cuanto es oprimido por segunda ocasión.

- setOnClickListener sobre el botón Switch Camara. - Al oprimirse el botón muestra en pantalla el contenido captado por cada cámara del dispositivo.

Una vez definidos los principales métodos y acciones del aplicativo, este puede ser compilado para generar un APK el cual provee un método de instalación del aplicativo en los dispositivos Android.

4.3.3 Pruebas módulo generador de contenido

Para la realización de pruebas de este módulo inicialmente se verificará**n** las conexiones hacia el servidor Ubuntu bajo el protocolo RMTP, mismo que trabaja en por defecto en el puerto 1935. La figura a continuación evidencia las conexiones establecidas con el servidor.

```
root@ubuntu:/video_recordings# netstat
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 192.168.1.37:1935       192.168.1.6:43912       ESTABLISHED
tcp        0      0 localhost:1935          localhost:36862         ESTABLISHED
tcp        0      0 localhost:36860         localhost:1935          ESTABLISHED
tcp        0      0 localhost:36862         localhost:1935          ESTABLISHED
tcp        0      0 localhost:1935          localhost:36860         ESTABLISHED
tcp        0    164 192.168.1.37:ssh        192.168.1.4:52849       ESTABLISHED
```

Figura 25. Conexiones establecidas en el servidor de decodificador/codificador.

En la figura 25 se puede apreciar que el servidor está realizando una conexión iniciada desde la ip 192.168.1.6 y cuyo puerto de destino es el 1935, es decir una conexión con el protocolo RTMP entre el servidor y la aplicación Android.
Como acto siguiente se debe verificar que la transmisión este siendo realizada de manera correcta, por tanto, podemos usar el aplicativo vod implementado en NGINX y visualizarlo mediante el reproductor multimedia VLC instalado en un dispositivo que se encuentre en el mismo segmento de red que el servidor Ubuntu y aplicativo Android, el contenido se desplegará automáticamente al ingresar el String de conexión para el video en específico. El gráfico siguiente muestra la visualización del video almacenado en el servidor.

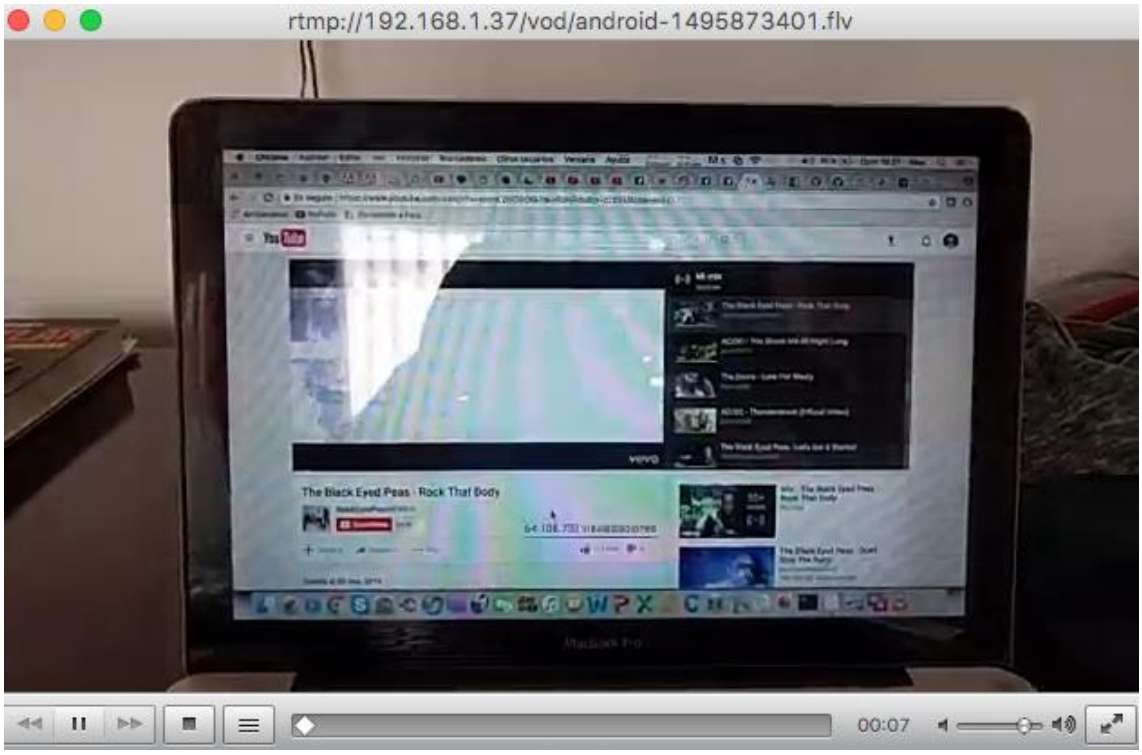


Figura 26. Visualización del streaming de video transmitido por el aplicativo Android.

Tal como se evidencia en la figura 26, el streaming de video se está transmitiendo de manera correcta, tras finalizada esta última sección se puede dar por concluida la fase de desarrollo del aplicativo por lo cual se debe dar paso a una fase de pruebas de todo el sistema.

4.4 Pruebas del sistema integrado y consumo de recursos

Habiendo finalizada la etapa de desarrollo del prototipo y las pruebas independientes de los módulos que conforman la solución, es de vital importancia la realización de pruebas de evaluación y consumo de recursos del sistema para identificación de los problemas y fortalezas de la plataforma, esto será de vital importancia en la decisión sobre la implementación a manera de proyecto del prototipo desarrollado.

4.4.1 Tiempo de retardo emisión - recepción de contenido

Dado que el servicio implementado está orientado a las transmisiones en vivo el retardo entre la captura del contenido y su visualización debe tener un retardo mínimo. Ya que los resultados obtenidos tras las encuestas realizadas a los estudiantes de la Universidad de las Américas reflejaron que el 87,5 % de los mismos usará su dispositivo celular para la visualización del contenido es natural prestar mayor atención a este grupo. Es de conocimiento general que los dispositivos móviles requieren una menor resolución para la reproducción de video debido al tamaño que presentan sus pantallas, por tal motivo el consumo del servicio a través de dispositivos móviles posee una menor resolución para mejorar el consumo de recursos de procesamiento y red.

Las pruebas de retardo se han realizado bajo un ambiente controlado con las siguientes características:

- Conexiones de red inalámbricas entre los componentes de la solución a través de un módem casero operando bajo el estándar 802.11g.
- Equipos dedicados para el funcionamiento de cada módulo.
- Aplicación cronometro para la medición del retardo.
- Dispositivo Android Nexus 5x para la emisión de contenido.
- Dispositivo móvil Samsung Galaxy 5s, laptop macbook pro y laptop Dell XPS 15, estos actuarán como clientes del sistema mediante el consumo de los recursos HLS resolución baja, HLS resolución estándar y RMTP resolución estándar respectivamente.

En la tabla 2 a continuación se da a conocer los resultados del tiempo de retardo obtenido entre la emisión y recepción del contenido, tras realizadas las pruebas con cada dispositivo mencionado anteriormente.

Tabla 2.

Resultados pruebas de retraso emisión – recepción de contenido.

Tiempo / Protocolo-Resolución	HLS baja resolución	HLS resolución estándar	RTMP resolución estándar
Medición 1 [s]	8,50	8,63	0,34
Medición 2 [s]	8,46	8,44	0,35
Medición 3 [s]	8,53	8,72	0,3
Promedio [s]	8,50	8,60	0,33

Tal como lo demuestran las pruebas el protocolo RTMP usado en laptops y computadores de escritorio tiene un mayor desempeño en cuanto a retardo se refiere, lamentablemente este no puede ser universalmente entendible en la totalidad de las plataformas debido a su dependencia con Flash Media Player. Por otra parte, el retardo de visualización existente entre dispositivos móviles y dispositivos Apple es similar ya que ambos usan el protocolo HLS para transmitir el contenido.

4.4.2 Consumo de Ancho de Banda

Un recurso que se debe tener en consideración es el Ancho de Banda consumido por las conexiones entre el servidor de codificación/decodificación y los clientes que requieren el contenido del mismo, debido a su capacidad limitada y el alto costo que genera el aumentarlo. Para realizar las mediciones de consumo de este recurso, se procedió a instalar el paquete iftop el cual se encargará de testear el ancho de banda consumido por las conexiones existentes entre los clientes y el servidor.

Para la realización de estas pruebas se ha medido el consumo de ancho de banda de los tres tipos de video que han sido configurados en el servidor de streaming y adicionalmente el consumo generado por la aplicación Android de manera independiente, por medio del software iftop instalado en el servidor de codificación/decodificación. La tabla 3 refleja los resultados obtenidos expresados en kilobits / segundo tras una medición en un intervalo de tiempo de un minuto.

Tabla 3.

Consumo de ancho de banda

Dispositivo	Aplicación Android	Móvil	Apple	Laptop/ PC
Ancho de Banda Kb/s	667	453	647	593

Una vez obtenidos los resultados de medición se puede constatar que el consumo de ancho de banda por medio de un dispositivo móvil es más bajo comparándolo con el resto de dispositivos, esto se debe al proceso de escalamiento de la resolución que es realizado por el servidor codificador/decodificador el cual da como resultado un streaming de video con menor peso, esto se ve reflejado con un menor consumo en el ancho de banda.

4.4.3 Consumo de recursos de procesamiento y memoria

En esta sección se analizarán los recursos provistos para el sistema, de esta manera se podrá concluir si existe un sobredimensionamiento o caso contrario una asignación insuficiente de los recursos. Para la obtención de los datos necesarios para el análisis se ha instalado el paquete htop el cual es capaz de presentar el consumo de recursos de procesamiento y memoria de manera gráfica sobre la consola provista en el sistema operativo, dado que el procesamiento de los videos se realizará solamente una vez de manera independiente al número de clientes que soliciten el mismo, la medición de los recursos utilizados se realizará en los casos en el que el servidor se encuentra procesando el streaming emitido y en el que se mantiene realizando únicamente tareas propias del sistema. En la figura siguiente se puede apreciar el consumo de recursos de procesamiento y memoria mientras el servidor realiza únicamente funciones propias del sistema operativo.

Figura 27. Consumo normal de procesamiento y memoria del servidor Ubuntu.

Tal como se puede apreciar en la figura 27, únicamente un procesador de un

total de cuatro está siendo utilizado con una carga mínima de trabajo, adicionalmente se puede constatar que el consumo de memoria RAM es relativamente bajo respecto al asignado como recurso para el servidor.

Una vez verificados los recursos en un estado normal del servidor, se realizará una transmisión con el fin de observar el comportamiento del sistema bajo los parámetros para los cuales este fue diseñado. La figura 28 señala el comportamiento de los recursos del servidor al realizar las funciones para las cuales fue implementado.

Figura 28. Consumo de procesamiento y memoria servidor Ubuntu durante transmisión de video en vivo.

Como se puede apreciar en el gráfico existe un sobredimensionamiento de recursos, en cuanto a la memoria RAM no se llega a ocupar más que el diez o quince por ciento del total asignado, por otro lado respecto al procesamiento únicamente se usan tres procesadores de los cuales dos tienen una carga mínima, se debe tener en consideración que estas pruebas se han realizado en un ambiente de desarrollo por lo cual podrían variar en una implementación en producción debido al número de peticiones que pueden ser hechas al módulo web.

5. CONCLUSIONES Y RECOMENDACIONES

5.1 Conclusiones

Los objetivos del proyecto están enfocados en brindar un servicio de video en vivo para la comunidad universitaria de la FICA de la UDLA mediante la implementación inicial de un prototipo compuesto principalmente por un servidor de multidifusión, un aplicativo Android para la generación de contenido y un aplicativo web para la visualización del mismo, debido a los resultados obtenidos tras realizadas las encuestas a un grupo aleatorio de la comunidad universitaria, el alcance fue extendido para poder cubrir un requerimiento de multiplataforma en cuanto a la visualización de los eventos emitidos.

Tras realizadas las pruebas modulares y globales se ha logrado constatar un correcto funcionamiento del prototipo implementado con respecto a la emisión y recepción del contenido, más sin embargo el tiempo de procesamiento introducido por el servidor NGINX bajo el protocolo HLS es superior a las expectativas que se tenía sobre este.

Durante las fases de análisis y desarrollo del prototipo se evidencia que los recursos de hardware y software necesarios para un correcto desempeño del prototipo son altamente convenientes debido al buen desempeño y gratuidad del software seleccionado para esta implementación.

5.2 Recomendaciones

Debido al esfuerzo invertido en el desarrollo del prototipo y el resultado positivo ante las pruebas funcionales del mismo, principalmente se recomienda la puesta en marcha de un ambiente de producción para prototipo, no sin antes sugerir la implementación adicional de las funcionalidades de balanceo de carga con un servidor NGIX adicional, lo cual se verá reflejado en un menor tiempo de espera para el consumo del servicio por parte del usuario final.

Como consecuencia del retardo inducido en las transmisiones haciendo uso del

protocolo HLS para dispositivos móviles y dispositivos Apple, se recomienda que la visualización de los eventos en vivo sea por medio de laptops y computadores de escritorio debido al retardo menor a un segundo que existe entre la emisión y recepción del contenido, argumentando además que el consumo del ancho de banda de las conexiones entre el servidor y los clientes es similar para todos los casos.

Para futuros proyectos se sugiere investigar sobre el costo beneficio de adquirir una licencia NGINX PLUS la cual permitirá entre otras cosas hacer uso del protocolo estándar de streaming DASH, mismo que además de ser universalmente usado en dispositivos, este puede proveer calidad de servicio a nivel de aplicativo.

REFERENCIAS

Arroyo, A., Navarro, C., López, J., González de Dios, J., & Aleixandre, R. (2015). Comunicación científica (XXVIII). Nuevas formas de difusión de contenidos: streaming, webcasting y podcasting. Recuperado el 20 de mayo de 2017 de http://pesquisa.bvsalud.org/bvsvs/resource/es/ibc-146553

Andro4all. (2013). Repaso a la evolución de las pantallas de los smartphones. Recuperado el 22 de Mayo del 2017, de https://andro4all.com/2013/01/repaso-a-la-evolucion-de-las-pantallas-de-los-smartphones

Backes, M., Bugiel, S., Derr, E., Mcdaniel, P., Octeau, D., & Weisgerber, S. (2016). On Demystifying the Android Application Framework: Re-Visiting Android Permission Specification Analysis. Recuperado el 17 de Abril del 2017, de https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/backes_android

Bai, F., Song, L., Tao, Y., y Liu, Y. (2013). Video Image Restoration Based on H.264 Protocol. In *2013 Third International Conference on Instrumentation, Measurement, Computer, Communication and Control* (pp. 500–503). IEEE. Doi 10.1109/IMCCC.2013.114

Begen, A., Akgul, T., & Baugher, M. (2011). Watching Video over the Web: Part 1: Streaming Protocols. IEEE Internet Computing, 15 (2), 54–63. Doi 10.1109/MIC.2010.155

Bouzakaria, N., Concolato, C., & Le, J. (2014). Overhead and performance of low latency live streaming using MPEG-DASH. In *IISA 2014, The 5th International Conference on Information, Intelligence, Systems and Applications* (pp. 92–97). IEEE. Doi 10.1109/IISA.2014.6878732

CISCO. (2000). Introduction IP Multicast. Recuperado el 22 de Mayo del 2017, de http://www.cisco.com/networkers/nw00/pres/2214.pdf

Du, Y., Liu, J., Liu, F., & Chen, L. (2014). A Real-Time Anomalies Detection System Based on Streaming Technology. In *2014 Sixth International Conference on Intelligent Human-Machine Systems and Cybernetics* (pp. 275–279). IEEE. Doi 10.1109/IHMSC.2014.168

Vásquez, E. (2014) *¿Qué es el bitrate de un video y para qué sirve?* Recuperado el 13 de Enero del 2017, de http://mentecuriosa.net/bitrate-video-sirve/

Garcia, S., Cabrera, J., y Garcia, N. (2014). Quality-optimization algorithm based on stochastic dynamic programming for MPEG DASH video streaming. In *2014 IEEE International Conference on Consumer Electronics (ICCE)* (pp. 574–575). IEEE. Doi 10.1109/ICCE.2014.6776139

García, C., Galán, J., y Izquierdo, R. (2017). *Multidisciplinary journal for education, social and technological sciences*. Recuperado el 11 de Diciembre del 2016, de https://polipapers.upv.es/index.php/MUSE/article/view/6347/7490

Geert, B. (2012). MIDI jitter might be ruining your live performance | Expressiveness. Recuperado el 05 de Enero del 2017, de http://expressiveness.org/2012/12/04/midi-jitter

Guanfeng, L., & Ben, L. (2008). Effect of Delay and Buffering on Jitter-Free Streaming Over Random VBR Channels. *IEEE Transactions on Multimedia*, *10*(6), 1128–1141. Doi 10.1109/TMM.2008.2001364

Hoque, M., Siekkinen, M., Nurminen, J., Tarkoma, S., y Aalto, M. (2014). Saving Energy in Mobile Devices for On-Demand Multimedia Streaming -- A Cross-Layer Approach. *ACM Transactions on Multimedia Computing, Communications, and Applications*, *10*(3), 1–23. Doi 10.1145/2556942

Hoßfeld, T., Schatz, R., y Krieger, U. (2014). QoE of YouTube Video Streaming for Current Internet Transport Protocols (pp. 136–150). Springer, Cham. Doi 1007/978-3-319-05359-2_10

Jestratjew, A., y Kwiecien, A. (2013). Performance of HTTP Protocol in Networked Control Systems. *IEEE Transactions on Industrial Informatics*, *9*(1), 271–276. Doi 10.1109/TII.2012.2183138

Keshavarz, A., y Riedi, R. (2014). Bounds on the Benefit of Network Coding for Wireless Multicast and Unicast. *IEEE Transactions on Mobile Computing*, *13*(1), 102–115. Doi 10.1109/TMC.2012.234

Kumar, S. (2014). UBIQUITOUS SMART HOME SYSTEM USING ANDROID APPLICATION. *International Journal of Computer Networks & Communications (IJCNC)*, *6*(1), 33–43. Doi 10.5121/ijcnc.2014.6103

Li, Y., y Liang, Y. (2016). Temporal Lossless and Lossy Compression in Wireless Sensor Networks. *ACM Transactions on Sensor Networks*, *12*(4), 1–35. Doi 10.1145/2990196

Liu, X., Wang, L., Wang, Z., Gao, X., y Zhu, M. (2015). An HD IPTV system based on scalable video coding. In *2015 IEEE International Conference on Digital Signal Processing (DSP)* (pp. 991–995). IEEE. Doi 10.1109/ICDSP.2015.7252026

Liu, Z., Wang, Q., Huang, J., Wu, Y., Wang, Y., Jia, X., y Chen, H. (2017). Cloud-based video-on-demand services for smart TV. In *2017 Seventh International Conference on Information Science and Technology (ICIST)* (pp. 81–84). IEEE. Doi 10.1109/ICIST.2017.7926496

Liu, Z., Wang, Q., Huang, J., Wu, Y., Wang, Y., Jia, X., y Chen, H. (2017). Cloud-based video-on-demand services for smart TV. In *2017 Seventh International Conference on Information Science and Technology (ICIST)* (pp. 81–84). IEEE. Doi 10.1109/ICIST.2017.7926496

Pattaranantakul, M., Sanguannam, K., Sangwongngam, P., y Vorakulpipat, C. (2015). Efficient Key Management Protocol for Secure RTMP Video Streaming toward Trusted Quantum Network. ETRI Journal, 37(4), 696–706. Doi 10.4218/etrij.15.0114.0883

Phil, H. (2008). Understanding Video Codecs | AvNetwork.com. Recuperado el 17 de Noviembre del 2017, de http://www.avnetwork.com/avtechnology/understanding-video-codecs/107681

Sanogo, H. (2010). A Proposed Framework for Measuring, Identifying, and Eliminating Clock and Data Jitter on High-Speed Serial Communication Links. Recuperado el 27 de Diciembre del 2016, de http://pdfserv.maximintegrated.com/en/an/AN4613.pdf

Seufert, M., Egger, S., Slanina, M., Zinner, T., Hobfeld, T., y Tran, P. (2015). A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys & Tutorials*, *17*(1), 469–492. Doi 10.1109/COMST.2014.2360940

Microsoft. (s.f.). Streaming en vivo con Azure Media Services para crear transmisiones con velocidad de bits múltiple. Recuperado el 3 de Febrero del 2017, de https://docs.microsoft.com/es-es/azure/media-services/media-services-manage-live-encoder-enabled-channels

Taleb, T., Ksentini, A., y Jantti, R. (2016). "Anything as a Service" for 5G Mobile Systems. *IEEE Network*, *30*(6), 84–91. Doi 10.1109/MNET.2016.1500244RP

Thang, T., Le, H., Pham, A. y Ro, Y. (2014). An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming. *IEEE Journal on Selected Areas in Communications*, *32*(4), 693–705. Doi 10.1109/JSAC.2014.140403

Thomas, E., Van, M., Begen, A., Famaey, J., y Stockhammer, T. (2015). Enhancing MPEG dash performance via server and network assistance. In IBC 2015 Conference (p. 8). Institution of Engineering and Technology. Doi 10.1049/ibc.2015.0014

Túqueres, C., 2015. Evaluación de la Red de Telefonía IP de la FIE para la Implementación del Uso de Smartphone. Recuperado el 16 de julio de 2017, de http://dspace.espoch.edu.ec/handle/123456789/4471

Vasanthi, V., y Chidambaram, M. (2250). A Study on Video Streaming in Cloud Environment. Recuperado el 27 de mayo de 2017 de https://pdfs.semanticscholar.org/8c49/324d13dd93062b6c314dfd7667b0d314e208.pdf

Vayavya (2014). Adaptive Streaming: MPEG DASH. Recuperado el 22 de Mayo del 2017, de http://vayavyalabs.com/solutions-ip/streaming-ip-integration-services/

Yadav, R., y Bhadoria, R. (2015). Performance Analysis for Android Runtime Environment. In *2015 Fifth International Conference on Communication Systems and Network Technologies* (pp. 1076–1079). IEEE. Doi 10.1109/CSNT.2015.52

Yang, G., Choi, B., y Kim, J. (2014). Implementation of HTTP Live Streaming for an IP Camera using an Open Source Multimedia Converter. *International Journal of Software Engineering and Its Applications*, *8*(6), 39–50. Doi 10.14257/ijseia.2014.8.6.04

Zhou, W., Xu, J. y Wang, B. (2017). A smart home foundation scheme based on open source hardware and cloud computing. *International Journal of Internet Protocol Technology*, *10*(1), 13. Doi 10.1504/IJIPT.2017.083032

Zingade, S., Joshi, R., Shendkar, R., Arora, P., y Scholar, U. (2016). DREAM – A Data Streaming Application Using RTP/RTSP in a Local Area Network. *International Journal of Engineering Science and Computing*, *6*(3), 2975–2977. Doi 10.4010/2016.693