TRANSMISIÓN DE VIDEO DE ALTA CALIDAD A TRAVÉS DE REDES IP UTILIZANDO HERRAMIENTAS DE CÓDIGO ABIERTO

Mauricio Venegas M., Aquiles Yáñez C., Agustín J. González
Departamento de Electrónica, Universidad Técnica Federico Santa María, Casilla 110-V, Valparaíso
[mauven, yanez, agv]@elo.utfsm.cl

Resumen

Este documento describe el proceso de captura de video en GNU/Linux y su transmisión sobre redes IP. Se detallan los requerimientos de hardware, configuraciones del sistema operativo y la instalación de programas específicos para transmitir un flujo de video a clientes remotos. Para lograr el objetivo es necesario utilizar dispositivos de captura compatibles con las APIs Video for Linux (*V4L*) para video, y OSS o ALSA para audio. Además estas APIs se utilizan en conjunto con bibliotecas y herramientas de software libre. Se describe y compara una herramienta basada en el proyecto MPEG4IP y la otra en el proyecto VIDEOLAN.

Palabras claves : Streaming, Transmisión de Video, V4L, V4L2, OSS, ALSA, MPEG4IP y VLC.

Abstract

This paper describes the process video capture process in GNU/Linux and its transmission on IP networks. It gives details on hardware requirements, operating system configuration, and the installation of specific programs for transmitting video streaming to remote clients. To achieve this goal, capture devices must be compliant with Video for Linux (V4L) and with OSS or ALSA for audio; in addition public libraries and tools are needed. We describe and compare one tool based on MPEG4IP project and another based on VIDEOLAN.

Key words : Streaming, Video Transmission, V4L, V4L2, OSS, ALSA, MPEG4IP y VLC.

1.- Introducción

Durante mucho tiempo el video ha sido un importante medio de comunicación y entretenimiento. Desde sus comienzos se utilizó tecnología analógica para su captura, transmisión, almacenamiento y reproducción. El ejemplo más representativo y que nos acompaña desde hace muchos años, es la televisión convencional, la cual es el principal medio de comunicación en la mayoría de los países. Sin embargo, con la llegada de la tecnología digital y la masificación de los computadores, la transmisión digital de contenido multimedia es el sucesor natural.

A. Problema general

La compresión de video y su distribución es un área de constante investigación y desarrollo desde fines de los 80'. A partir del desarrollo de estas técnicas, existen hoy en día una gran variedad de aplicaciones para video digital tales como: VCD, DVD, DVB, y DTV [1].

La popularidad y crecimiento de Internet ha motivado el empleo de redes IP para la transmisión de video. Sin embargo, la baja calidad de video que se consigue en las transmisiones en vivo es un obstáculo a superar.

Teniendo en cuenta lo anterior se puede considerar que es posible migrar a redes IP algunos sistemas de video analógicos (tanto abiertos como cerrados) teniendo en consideración los siguientes motivos:

- En general, existe un alto costo asociado a los equipos utilizados en la transmisión de video analógico.
- En algunos escenarios, es necesaria la autorización para utilizar espectros de radiofrecuencia.
- Las redes IP presentan una arquitectura de mayor flexibilidad y con una cobertura en crecimiento.
- Con las tecnologías emergentes surgen nuevos conceptos, como IPTV [1] y Streaming.
- Hoy en día existe un importante auge en el uso de sistemas operativos y herramientas de código abierto.

Con soluciones económicas para transmitir video, muchas actividades pueden ser favorecidas, como por ejemplo: telemedicina, educación a distancia, video vigilancia, entretenimiento, presentaciones y conferencias.

B. Conceptos básicos sobre Streaming

Streaming es la transferencia de contenido audiovisual por la red desde un servidor hacia sus clientes, con la característica de que es posible visualizar el contenido en la medida que el flujo es recibido. Antes era necesario descargar completamente el archivo de video pare poder reproducirlo, lo que implicaba un gran tiempo de espera. Para comprimir el video se utilizan codecs y formatos contenedores especiales para permitir streaming. En la Figura 1 se aprecian las principales etapas involucradas en el proceso de Streaming, a saber:

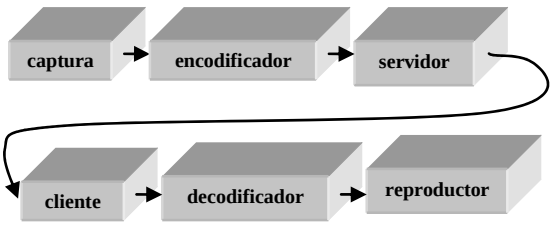

Fig.1. Etapas conceptuales dentro del proceso de Streaming.

- Captura: El contenido audiovisual es capturado y convertido en datos digitales crudos, aún sin codificación y compresión.
- Encodificador: Los datos crudos de la etapa anterior que contienen gran cantidad de redundancia espacial y temporal, son codificados por algún codec específico y son encapsulados por algún formato contenedor particular.
- Servidor: El flujo continuo de datos es distribuido por la red a los diferentes clientes ya sea a través de unicast, multicast o broadcast.
- Cliente: En esta etapa se reciben los datos desde la red y se envían como un flujo a la etapa de decodificación.
- Decodificador: En esta etapa los datos son descodificados y descomprimidos de acuerdo al formato contenedor y a los codecs usados en el proceso de encodificación.
- Reproductor: Esta es la etapa en que el contenido multimedia regenerado a partir de los datos es desplegado en pantalla y reproducido por la tarjeta de sonido del receptor.

C. Adquisición de audio y video

Un aspecto muy importante en el proceso de streaming es la adquisición de audio y video desde los dispositivos de captura. Esta etapa depende mucho del sistema operativo que se esté ocupando. En particular, depende de que existan en él, los drivers del dispositivo de captura. Los drivers se comunican con las aplicaciones mediante las interfaces de programación de aplicaciones (API`s). Los dispositivos específicos de captura, utilizados en este trabajo son los siguientes:

- Cámaras USB (Webcams): Se utilizó el dispositivo Phillips TouCam Pro II USB. El driver con que opera es pwd y se compiló especialmente para el kernel Linux 2.6.x. Esta cámara sólo puede capturar video a 640x480@15 fps ó a 320x240@30 fps.
- Tarjeta capturadora de video (chipset bt878): Esta tarjeta capturadora funciona con el driver bttv en los kernel Linux versión 2.6 y permite capturar video a 640x480@30 fps.

Además, se utilizaron las siguientes interfaces de programación de aplicaciones (API´s):

- Video for Linux (V4L) [2]: Esta API está orientada netamente a la captura de vídeo para plataformas GNU/Linux. Fue incluida en el kernel Linux desde la versión 2.4 y soporta una amplia gama de dispositivos de captura.
- Video for Linux 2 (V4L2) [2]: Ésta es una versión mejorada de la API Video for Linux, y apareció en el kernel Linux desde la versión 2.5. Esta versión se orientó tanto a la captura de video como a la reproducción, esto es útil para el caso de los dispositivos que poseen salidas de video. Esta nueva versión es mucho más robusta, completa y compleja que la versión anterior. Video for Linux 2 tiene módulos de compatibilidad hacia atrás.
- Open Sound System (OSS) [3]: Ésta es una API orientada a la captura y reproducción de sonido. Era la API que se incluía en el kernel Linux desde la versión 2.4 y anteriores. Esta API soporta una gran cantidad de dispositivos de hardware tales como tarjetas de sonido y chipset de audio incluidos en placas madres.
- Advanced Linux Sound Architecture (ALSA) [4]: Es la sucesora de la API OSS, en los kernel Linux de la serie 2.6 en adelante. ALSA también tiene módulos de compatibilidad con aplicaciones programadas en OSS.

2.- Diseño e Implementación

Para implementar una solución de streaming es optó por menos dos herramientas basadas en proyectos de código abierto, además de contar con los codecs para audio y video. En la arquitectura de red de pruebas todas las máquinas tienen instalado el sistema operativo Debian GNU/Linux 3.1 [5].

A. Plataformas propuestas

Después de estudiar varios proyectos de streaming, se optó por 2 soluciones que poseían las condiciones necesarias para cumplir con los objetivos tanto de calidad como de licencias. La primera está basada en el proyecto VideoLAN [6], el cual es una solución integral para el streaming de video y que posee licencia GNU (GPL). La segunda plataforma está basada en el proyecto de código abierto MPEG4IP [7]. Este proyecto originalmente comenzó a ser desarrollado por CISCO, pero posteriormente paso a formar parte de la comunidad de código abierto internacional.

Cada una de estas suites, se configuró para usar 2 grupos de codecs:

- MPEG-2 (video) con MP3 (audio): Se escogió este par de codecs porque son bastante utilizados en la actualidad y su buen soporte en ambos proyectos de streaming.
- MPEG-4 (video) con AAC (audio): Se escogieron porque son más nuevos.

B. Red de pruebas y software utilizado

Como entorno de pruebas se ocupó una red Ethernet donde todos los hosts estaban conectados a través de un Hub de 10/100 Mbps. El diagrama de esta red se puede apreciar en la figura 2. Se prefirió usar un Hub porque permite un fácil monitoreo de la red. En cuanto a las máquinas dispuestas, se utilizó un computador principal, encargado de generar el flujo y que denominamos servidor, y otros dos PC's, cliente 1 y cliente 2, que sólo reciben el flujo. Además hay una cuarta máquina, "analizador", que sólo se encarga de monitorear la red, y que no interviene en el proceso de streaming directamente. La configuración se muestra en Figura 2.

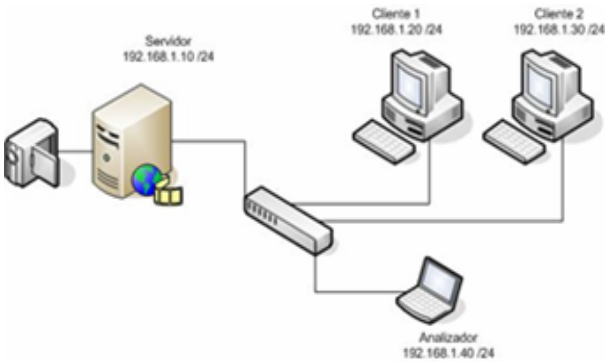

Fig.2. Diagrama esquemático de la red de pruebas utilizada.

El servidor posee un procesador Athlon XP 2100+, 512 Mb de memoria RAM, tarjeta capturadora con chipset bt878 y chip de sonido vt8235. El cliente 1 posee un procesador Pentium III de 450 Mhz, 128 Mb de memoria RAM, Tarjeta de sonido ES1370 y una tarjeta de video Riva TNT. Se escogió una máquina con recursos limitados para probar el desempeño de los programas clientes de ambas soluciones en este escenario. El cliente 2 posee un procesador Athlon XP 2000+, 512 MB de memoria RAM, chipset de audio vt8235 y tarjeta de video Riva TNT 2. Se escogió una maquina con estas características, por su similitud a las máquinas disponibles en el mercado, a la fecha de realización de estas pruebas. La máquina denominada analizador corresponde a un Laptop Toshiba A70.

Otro aspecto importante es el software a utilizar. Las bibliotecas que se instalaron en el sistema son las siguientes:

- Bibliotecas de Codecs de audio: MPEG Layer 3 (libLAME [8], libMAD [9]) y AAC (libfaac, libfaad2 [10]).
- Bibliotecas de Codecs de video: MPEG-2 (libmpeg [11]), MPEG-4 (xvid [12]).
- Bibliotecas de streaming: UCL [13], liveMedia [14], libdvbpsi [15].
- Bibliotecas de despliegue: libSDL [16].

Como tópico aparte se considera la biblioteca libavcodec, del proyecto FFMPEG, la cual también es usada por otras aplicaciones multimedia por su gran cantidad de codecs para audio y video.

A continuación se nombraran tanto los programan servidores de cada proyecto como sus respectivos reproductores en las máquinas cliente.

- Servidores de streaming: Mp4Live (MPEG4IP), VLC (VideoLAN).
- Reproductores de multimedios: Mplayer (MPEG4IP), VLC, Gmp4player (MPEG4IP).

C. Proyecto VideoLAN

Este proyecto es una solución completa de software para el streaming de video, desarrollada bajo la licencia pública general GNU (GPL). VideoLAN está diseñado para generar flujos de videos bajo el estándar MPEG sobre redes de alto ancho de banda. Soporta el transporte de datos a través de los protocolos IPv4 e IPv6 además de poder generar tráfico unicast o multicast.

Dentro de las herramientas desarrolladas por el proyecto VideoLAN, se encuentran vls (VideoLAN Server) y vlc (inicialmente el cliente VideoLAN). A pesar de que partieron siendo programas complementarios (cliente y servidor), con el transcurso del tiempo, el desarrollo de vlc permitió que esta herramienta fuera una solución en si misma. Es por ello que nuestra atención estará centrada en esta aplicación.

El programa vlc permite que se opere a través de línea de comandos, o bien, por su interfaz gráfica. Se recomienda, sin embargo, la utilización en línea de comandos ya que permite mayor control sobre los parámetros con los cuales se invoca al servidor. Es importante destacar que, las herramientas de VideoLAN sobresalen sobre los otros sistemas de streaming ya que poseen una excelente documentación y una interfaz más amistosa, lo que permite una rápida adaptación por parte de usuarios menos experimentados.

D. Proyecto MPEG4IP

Este proyecto provee un conjunto de herramientas de código abierto que permiten implementar servidores y clientes de streaming basados en estándares exentos de protocolos y extensiones propietarios. Su desarrollo está focalizado a GNU/Linux, pero también ha sido portado a otras plataformas. Dentro de los muchos programas que se incluyen en el proyecto se encuentran el reproductor gmp4player y el servidor mp4live. Éste permite codificar y transmitir flujos de video y audio en tiempo real, obtenidos a través de la captura desde dispositivos compatibles con V4L.

El servidor mp4live dispone de una Interfaz Gráfica de Usuario (GUI) que permite configurar los parámetros necesarios para generar el streaming. La configuración puede ser también leída desde un archivo llamado .mp4live_rc. Para aumentar la cantidad de codecs soportados por esta herramienta es posible aplicarle parches al código fuente. De todas maneras existe soporte nativo para el formato de video MPEG-4 y para los formatos de audio MP3 y AAC. El resultado de la captura puede ser transmitido por la red vía unicast o multicast, escrito localmente a un archivo .mp4 o

ambos simultáneamente. En el caso de ser transmitido por la red, se utiliza el protocolo RTP [17] sobre datagramas UDP [18].

3.- Evaluación y Análisis

El trabajo de comparar las alternativas de software libre para transmitir video, requirió la aplicación de métodos cuantitativos y, en menor medida, métodos cualitativos. A continuación se pasa a describir los criterios y herramientas utilizadas en la evaluación.

A. Criterios y herramientas de evaluación

El principio utilizado para conseguir la evaluación de cada solución de streaming, fue obtener índices internos del sistema que pudiesen entregar información relevante, al momento de generar una transmisión conjunta de video y audio. Los criterios de medición utilizados se pueden separar en dos: medidas de requerimientos en cada máquina y aquellas que corresponden al nivel de ocupación de la red.

El requerimiento en cada equipo fue medido a través del porcentaje de utilización de CPU y la cantidad de memoria de sistema. Para ello se desarrolló un programa en lenguaje C que lee las estadísticas de kernel Linux desde el directorio /proc. Este programa se denomina medidor en Figura 3.

El nivel de ocupación de la red fue obtenido a través de un script en Perl que utiliza un filtro para discriminar la información que entrega el programa tcpdump en tiempo real. Este programa se denomina monitor de streaming en Figura 3.

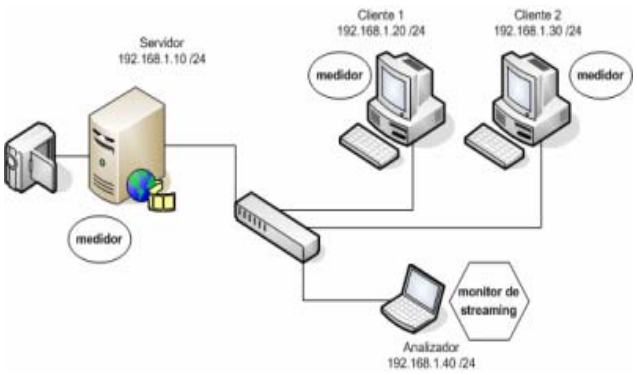

Fig.3. Disposición de las herramientas de medición en el sistema. Tanto servidor como clientes ejecutan el programa medidor, en cambio, el monitor de streaming se ejecuta en un computador externo.

B. Mediciones y análisis de los datos obtenidos

En total 18 mediciones fueron realizadas en las cuales se evaluó la capacidad de procesamiento de cada máquina involucrada además del nivel de utilización de la red. Cada una de ellas tuvo una duración de 300 segundos. La transmisión se comenzó a los 10 y se terminó a los 290 segundos aproximadamente. El objetivo de dejar algunos segundos de resguardo fue poder tener un intervalo de tiempo para apreciar el cambio en los niveles de procesamiento de cada máquina al ejecutar los programas de streaming.

Los valores promedio obtenidos al utilizar la herramienta de VideoLAN con codecs MPEG-2 y MP3, se pueden observar en las Tablas I.A y I.B.

TABLA I .A
A: PORCENTAJE DE USO DE CPU

	servidor	cliente 1	cliente 2
unicast rtp	84,00	73,59	-
unicast rtp	82,11	-	36,50
broadcast rtp	84,07	74,87	36,81
multicast udp	84,93	73,44	36,41
multicast rtp	83,33	73,93	36,50

TABLA I.B
TRÁFICO NIVEL IP EN KBITS/S

	ancho de banda
unicast rtp	4101,10
unicast rtp	4100,49
broadcast rtp	4102,17
multicast udp	4066,03
multicast rtp	4100,24

Los valores promedio obtenidos al utilizar la herramienta de VideoLAN con codecs MPEG-4 y AAC, se pueden observar en las Tablas II.A y II.B.

TABLA II.A
PORCENTAJE DE USO DE CPU

	servidor	cliente 1	cliente 2
unicast rtp	94,77	94,56	-
unicast rtp	95,18	-	40,87
broadcast rtp	97,80	95,38	41,13
multicast udp	97,98	95,31	40,73
multicast rtp	98,24	93,80	40,39

TABLA II.B
TRÁFICO NIVEL IP EN KBITS/S

	ancho de banda
unicast rtp	4072,33
unicast rtp	4073,13
broadcast rtp	4073,79
multicast udp	4035,71
multicast rtp	4074,53

Los valores promedio obtenidos al utilizar la herramienta de MPEG4IP (Servidor Mp4live y cliente Gmp4Player), con codecs MPEG-2 para video y MP3 para audio, se presentan en la Tabla III.A y III.B.

TABLA III.A
PORCENTAJE DE USO DE CPU

	servidor	cliente 1	cliente 2
unicast rtp	81,67	91,99	-
unicast rtp	79,29	-	53,34
multicast rtp	77,11	90,16	53,24

TABLA III.B
TRÁFICO NIVEL IP EN KBITS/S

	ancho de banda
unicast rtp	3333,22
unicast rtp	3342,53
multicast rtp	3347,40

Los valores promedio obtenidos al utilizar la herramienta de MPEG4IP (Servidor Mp4live y cliente Gmp4Player), con codecs MPEG-4 para video y AAC para audio, se presentan en la Tabla IV.A y IV.B.

TABLA IV.A
PORCENTAJE DE USO DE CPU

	servidor	cliente 1	cliente 2
unicast rtp	81,10	98,80	-
unicast rtp	84,98	-	63,41
multicast rtp	86,01	98,58	63,28

TABLA IV.B
TRÁFICO NIVEL IP EN KBITS/S

	ancho de banda
unicast rtp	3298,98
unicast rtp	3323,59
multicast rtp	3304,96

4.- Análisis Comparativo

A partir de los datos adquiridos se pudo generar una serie de comparaciones entre mediciones. Las Tablas V y VI representan la comparación de ancho de banda entre VLC y Mp4live, basado en los valores obtenidos de tablas anteriores.

TABLA V: CONSUMO DE ANCHO DE BANDA DE VLC V/S MP4LIVE PARA MPEG-2 CON MP3,TRÁFICO NIVEL IP EN KBITS/S

	Ancho de banda	
	Multicast	Unicast
VLC	4100,24	4100,79
MP4Live	3347,40	3337,87

TABLA VI: CONSUMO DE ANCHO DE BANDA DE VLC V/S MP4LIVE PARA MPEG-4 CON AAC, TRÁFICO NIVEL IP EN KBITS/S

	Ancho de banda	
	Multicast	Unicast
VLC	4074,53	4072,73
MP4Live	3304,96	3311,28

Se puede apreciar que la tasa en bit por segundos en una transmisión usando MP4Live es menor que con VLC, debido a que esta última agrega mayor overhead al ocupar el formato MPEG-TS [19].

En cuanto al consumo de recursos de CPU, los siguientes gráficos comparativos señalan la carga promedio en el servidor al utilizar VLC y Mp4live.

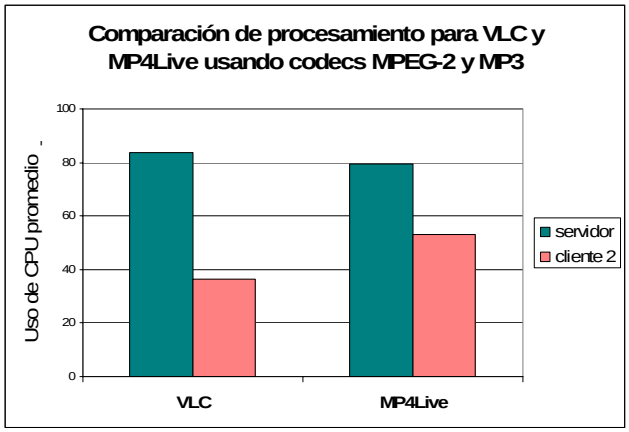

Fig.4. Gráfico comparativo de carga promedio en servidores VLC y Mp4live usando los codecs MPEG-2 y MP3.

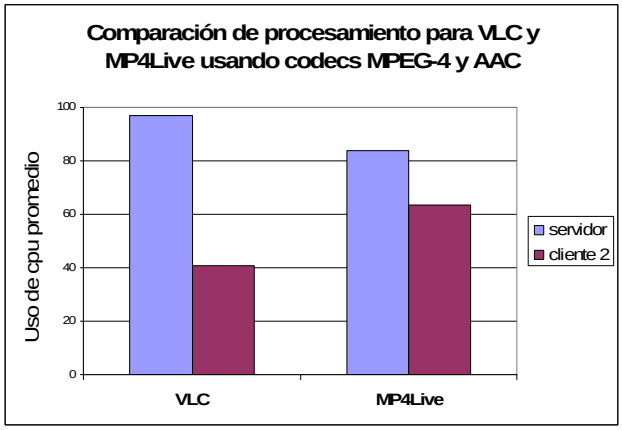

Fig.5. Gráfico comparativo de carga promedio en servidores VLC y Mp4live usando los codecs MPEG-4 y AAC.

Los gráficos de las figuras 5 y 6 señalan que, en nuestro sistema de streaming, fue más intensivo en cómputo la utilización de los códecs MPEG-4 y AAC, ya que utilizan algoritmos más complejos.

En Tablas VII y VIII se señalan los resultados obtenidos con cámaras USB (webcams) al utilizar el servidor y cliente VLC con codecs MPEG-4 y AAC. Los resultados obtenidos fueron no satisfactorios debido a que no se consiguieron los mínimos requerimientos de calidad en el video. Sin embargo, hay algunas ideas que se pueden obtener del estudio de estos datos.

TABLA VII.A: USO DE CPU DE VLC CON CODECS MPEG-4 Y AAC UTILIZANDO WEBCAM, PORCENTAJE DE USO DE CPU

	servidor	cliente 2
640x480@15fps	62,84	22,87
320x240@30fps	39,99	14,33

TABLA VII.B: CONSUMO DE ANCHO DE BANDA DE VLC CON CODECS MPEG-4 Y AAC UTILIZANDO WEBCAM, TRÁFICO NIVEL IP EN KBITS/S

	ancho de banda
640x480@15fps	2146,12
320x240@30fps	1951,82

Según estos números, se demuestra que es más intensivo para el procesador manipular resoluciones más grandes que aumentar la tasa de cuadros por segundos. El ancho de banda también se ve disminuido para el segundo caso.

Es de esperar que en un futuro cercano, existan webcams de bajo costo con las características de calidad suficientes para aplicarlas a un sistema de alta calidad de video.

5.- Conclusiones

A través del estudio de dos proyectos de código abierto para la transmisión de video (VideoLAN y MPEG4IP), se logró implementar una plataforma de streaming en vivo de alta calidad (640x480@30) sobre redes IP. Ésta se hizo operar usando computadores personales y hardware de captura de bajo costo. Lo anterior permite establecer soluciones económicamente atractivas para poderosos sistemas de streaming.

Para realizar un análisis comparativo entre las distintas configuraciones fueron desarrollados dos programas de medición. Éstos permitieron obtener datos cuantitativos como uso de cpu de las máquinas y ancho de banda utilizado.

Ambos servidores tratados (VLC y Mp4Live) son 2 soluciones de streaming de código abierto que presentaron un buen desempeño. Cada uno de ellos utilizan distintos enfoques de compresión y transmisión, lo que permite plantear distintas variantes para un mismo sistema.

Los resultados obtenidos permiten determinar las características mínimas de hardware para un buen desempeño considerando los complejos algoritmos para compresión de video y audio.

La elección de los grupos de codecs que se implementaron para cada servidor de streaming, influye en la calidad de la señal y en la capacidad de procesamiento requerida. Los factores analizados (CPU y BW) dependen tanto de la implementación del servidor como de los codecs elegidos.

Es fundamental la compatibilidad del equipamiento de captura, como cámaras USB y tarjeta capturadora de video, con las API's de video.

Referencias

[1] Definición de IPTV, http://en.wikipedia.org/wiki/IPTV.

[2] Video for Linux & Video for Linux 2, http://linux.bytesex.org/v4l2/.

[3] Open Sound System (OSS), http://www.opensound.com.

[4] Proyecto ALSA, http://www.alsa-project.org.

[5] Debian GNU/Linux, http://www.debian.org.

[6] Proyecto VideoLAN, http://www.videolan.org.

[7] Proyecto MPEG4IP, http://mpeg4ip.sourceforge.net.

[8] The LAME Project, http://lame.sourceforge.net.

[9] MPEG Audio Decoder, http://www.underbit.com/products/mad/.

[10] AudioCoding, http://www.audiocoding.com.

[11] Free MPEG-2 video stream decoder, http://libmpeg2.sourceforge.net.

[12] Codec XviD.org, http://www.xvid.org.

[13] UCL common multimedia library, http://www-mice.cs.ucl.ac.uk/multimedia/software/common/.

[14] LIVE.COM http://www.live.com/liveMedia/.

[15] Libdvbpsi (MPEG-TS), http://developers.videolan.org/libdvbpsi/.

[16] Simple DirectMedia Layer (SDL) http://www.libsdl.org.

[17] Real-time Transport Protocol (RTP), RFC-3550.

[18] User Datagram Protocol (UDP), RFC-768.

[19] MPEG-TS, http://erg.abdn.ac.uk/research/future-net/digital-video/mpeg2-trans.html.

Reseñas Biográficas

Mauricio Venegas M. obtuvo su título de Ingeniero Civil Electrónico en la Universidad Técnica Federico Santa María (UTFSM) el año 2005. Actualmente se desempeña en Adexus.

Aquiles Yáñez C. obtuvo su título de Ingeniero Civil Electrónico en la UTFSM el año 2005. Actualmente se desempeña Ingeniero de Plataforma de Altavoz.

Agustín J. González es Ingeniero Civil Electrónico de la UTFSM, tiene un magister en Ingeniería Electrónica de la misma universidad. Luego el señor González recibió los grados de Magister y Doctor en Ciencias de la Computación en la Old Dominion University, EEUU, en 1997 y año 2000. Actualmente se desempeña como Profesor Auxiliar del Departamento de Electrónica de la UTFSM.

Agradecimientos

Agradecemos el apoyo recibido de la Universidad Técnica Federico Santa María a través del proyecto USM 23.04.26.