Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

Set-top box personalizado para IPTV, utilizando herramientas de código abierto

Autor: José Antonio Fontaine González

Tutor: Ing. Rolando Pérez Versón

Santa Clara

2012

"Año 54 de la Revolución"

Universidad Central “Marta Abreu” de Las Villas

Facultad de Ingeniería Eléctrica

Departamento de Telecomunicaciones y Electrónica

TRABAJO DE DIPLOMA

Set-top box personalizado para IPTV, utilizando herramientas de código abierto

Autor: José Antonio Fontaine González

e-mail: jfontaine@uclv.edu.cu

Tutor: Ing. Rolando Pérez Versón

e-mail: rpverson@uclv.edu.cu

Consultante: MsC. Hiram del Castillo Sabido

Santa Clara

2012

"Año 54 de la Revolución"

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

Firma del Autor

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

Firma del Tutor

Firma del Jefe de Departamento donde se defiende el trabajo

Firma del Responsable de Información Científico-Técnica

PENSAMIENTO

Las personas lo suficientemente locas
como para pensar que pueden cambiar el
mundo son las que lo cambian.
Anuncio «Piensa diferente» de Apple,

1997

DEDICATORIA

A mi familia

A mi novia

A los buenos amigos

AGRADECIMIENTOS

A mi abuela por todo su cariño y dedicación.

A mis padres por sus esfuerzos para formar un hombre de bien.

A Gisela y Antonio por acogerme como un hijo más.

A Maday, por su amor y enorme paciencia durante este período.

A mi tutor Ing. Rolando Evelio Pérez Versón, por su apoyo.

A todo el que puso un grano de arena e hizo posible la realización de este trabajo.

Muchas Gracias.

TAREA TÉCNICA

1. Revisión bibliográfica y estudio de los trabajos relacionados con los sistemas de IPTV.
2. Análisis crítico de las herramientas que se puedan utilizar para el desarrollo del trabajo y el logro de los objetivos propuestos.
3. Creación del sistema propuesto con las diferentes herramientas seleccionadas.
4. Evaluación de la efectividad del sistema propuesto.
5. Confección del informe del trabajo de diploma.

Firma del Autor

Firma del Tutor

RESUMEN

La presente investigación estuvo enfocada en estudiar los aspectos fundamentales de un sistema de IPTV. Se realizó con el objetivo de implementar un *set-top box* personalizado, que permita brindar servicios relacionados con esta tecnología, utilizando para ello las ventajas que ofrece el universo del Software Libre. Para el logro de tal objetivo se efectuó una evaluación de las herramientas de código abierto disponibles, seleccionando la óptima para ser modificada y adaptada a las necesidades reales. Los principales resultados demostraron que el software seleccionado, luego de ser modificado, cumple con todos los requerimientos necesarios para la implementación del *set-top box*. Es decir, dicho software podrá ser instalado en un hardware de regulares prestaciones, para convertirlo en un dispositivo capaz de brindar servicios tales como: contenidos multimedia bajo demanda, televisión en vivo y navegación web. Con el producto obtenido se pretende contribuir a la implementación de nuevos servicios en las instalaciones hoteleras de la Cayería Norte de la provincia cubana de Villa Clara, con una reducción considerable de los costos económicos.

TABLA DE CONTENIDOS

INTRODUCCIÓN

Las instalaciones hoteleras existentes en la Cayería Norte de la provincia de Villa Clara (Cuba), son reconocidas a nivel mundial por su confort y calidad de servicios, sin embargo comparadas con instituciones similares en el resto del mundo, hay aspectos que todavía pueden mejorarse.

Servicios tan comunes como el *video on demand*, la televisión digital y la navegación por Internet, se encuentran completamente ausentes o se brindan de forma aislada, sin ningún tipo de estandarización, factor que incide negativamente en la satisfacción de clientes cada vez más exigentes.

La empresa TeleCable Internacional se ha propuesto la tarea de implementar estos nuevos servicios, utilizando una tecnología con gran aceptación y extensión a nivel global: IPTV.

IPTV describe los mecanismos para transportar contenidos multimedia sobre una red IP, y se ha seleccionado porque permite el aprovechamiento de la infraestructura ya instalada en la Cayería Norte, mediante la cual se distribuye actualmente la televisión por cable.

En el mercado existen sistemas completos de IPTV, que permiten el disfrute de los servicios relacionados con esta tecnología, distribuidos por varios fabricantes, pero sus precios hacen que la inversión no sea viable, teniendo en cuenta la situación económica de Cuba. Por tal motivo, la empresa TeleCable Internacional ha solicitado el apoyo de la UCLV, con el propósito de encontrar alternativas a tan costosos equipamientos, que permitan brindar los servicios deseados con una calidad óptima.

Para dar respuesta a tal exigencia el presente trabajo se ha destinado al desarrollo de un *set-top box* de IPTV personalizado, que no dependa de un sistema en específico para su

funcionamiento, lo cual contribuirá a la utilización de los recursos disponibles ya instalados.

La implementación de este *set-top box*, solo supondrá una pequeña inversión en la adquisición del hardware, pues para desarrollar el componente de software necesario, se utilizarán herramientas de código abierto libres de costo.

La búsqueda de soluciones prácticas que puedan ser utilizadas en entornos reales es una de las principales tareas que se traza cualquier institución universitaria que forme personal científico-técnico. La presente investigación está sustentada en dar respuesta al siguiente problema científico:

- ¿Cómo desarrollar un set-top box para IPTV utilizando software de código abierto?

Con este fin el objetivo general consiste en:

- Desarrollar un *set-top box* para IPTV, que presente una interfaz agradable y sencilla para el usuario, con el menor costo económico posible.

Para el logro de este objetivo general, se han planteado los siguientes objetivos específicos:

- Analizar los fundamentos teóricos de la tecnología IPTV.
- Seleccionar el software *Open Source* óptimo para el desarrollo del trabajo.
- Modificar el software seleccionado para que cumpla los requerimientos de los servicios deseados.
- Implementar en el software seleccionado un servicio de navegación web.
- Comprobar el producto terminado en un escenario real para validar su correcto funcionamiento.

La presente investigación pretende iniciar la creación de un sistema IPTV cubano, que no dependa de fabricantes externos y pueda personalizarse en función de las necesidades propias de las instituciones cubanas. Su valor práctico radica en brindar una herramienta mediante la cual podrán ofertarse nuevos servicios en las instalaciones hoteleras de la Cayería Norte, relacionados con la difusión de contenidos multimedia sobre la red HFC

existente en la zona. El alcance se relaciona con su utilidad para ser aplicado en otros ambientes donde existan redes IP.

Estructura del Informe.

Para dar cumplimiento a los objetivos propuestos, se ha organizado el contenido de la investigación de la siguiente forma:

- **Introducción**, en la cual se ofrece una panorámica de la problemática.
- **Primer Capítulo**, en el cual se realiza la descripción general de un sistema IPTV, brindando una breve panorámica de los módulos que lo conforman.
- **Segundo Capítulo**, el cual contiene una breve descripción de los principales módulos que se utilizaron para crear un *set-top box* para IPTV, utilizando aplicaciones de código abierto.
- **Tercer Capítulo**, el cual está dedicado a la descripción de las pruebas experimentales realizadas para validar el correcto funcionamiento del producto terminado, en un escenario real, utilizando herramientas de análisis histórico de datos en redes IP.
- **Conclusiones**, donde se exponen las consideraciones finales sobre la problemática investigada.
- Las **Recomendaciones**, que solicitan la profundización y ampliación de los estudios sobre la temática.
- Las **Referencias Bibliográficas**, que dan origen a la conformación del cuerpo investigativo.
- Los **Anexos**, que dan crédito y validez a los contenidos tratados en la investigación.
- **Glosario**, donde se aclara el significado de algunas palabras o siglas que puedan resultar difíciles de comprender para el lector.

CAPÍTULO 1. INTRODUCCIÓN A LA TECNOLOGÍA IPTV

En este capítulo se presentan los principales conceptos relacionados con el tema de la transmisión de contenidos multimedia sobre el protocolo IP. Se brinda una breve panorámica de los módulos que conforman un sistema diseñado con este fin y de los soportes de red utilizados típicamente para su implementación. Se tratan además, los conceptos fundamentales del *video streaming* por la importancia que estos suponen para el desarrollo del presente trabajo.

1.1 Características fundamentales

IPTV (*Internet Protocol Television*) también conocida como Telco TV o TV de banda ancha se trata de transmitir de forma segura y con alta calidad los canales tradicionales de Televisión Digital, video bajo demanda y contenido de audio sobre redes de banda ancha.(Kozamernik y Vermaele, 2005)

La Unión Internacional de Telecomunicaciones reconoce como definición oficial la siguiente:

> *IPTV está definida como un grupo de servicios multimedia tales como televisión/video/audio/texto/gráficos/datos difundidos sobre redes basadas en el protocolo IP, administradas para proveer los niveles requeridos de calidad de servicios y experiencia, interactividad y fiabilidad.*

En este epígrafe, se describen algunos de los aspectos fundamentales de esta tecnología tales como: ventajas fundamentales, principales servicios y estructura básica de un sistema creado para distribuirlos.

1.1.1 Ventajas

La tecnología IPTV constituye una de las más extendidas en el mundo para la distribución de contenidos digitales, debido a las ventajas que brinda con respecto a otras plataformas de transmisión. Destacándose como principales ventajas:

Soporte para Televisión Interactiva: las capacidades de comunicación bidireccional de los sistemas de IPTV permiten a los proveedores de servicios distribuir un amplio paquete de aplicaciones de TV Interactiva. No solo soportan la TV en vivo estándar y la TV de alta definición (*HDTV*), también permiten la inclusión de juegos interactivos y un rápido acceso a Internet. (García, 2004)

Desplazamiento en el tiempo (time shifting): La integración de un equipo de grabación digital al sistema IPTV permite el almacenamiento de programas para su posterior reproducción, así como el control de desplazamiento temporal. (Held, 2007)

Personalización: La bidireccionalidad de los sistemas IPTV permite además, que los usuarios finales personalicen sus hábitos de disfrutar la TV, decidiendo qué contenidos quieren ver y cuándo. (Erdogan, 2004)

Aprovechamiento eficiente del ancho de banda: En lugar de difundir cada canal a cada usuario, IPTV permite a los proveedores de servicios transmitir solamente el canal solicitado por el usuario. Esto garantiza un mejor aprovechamiento del ancho de banda de sus redes. (Hjelm, 2008)

1.1.2 Descripción general de un sistema IPTV

Un sistema de IPTV (Figura 1.1), puede representarse básicamente por tres bloques fundamentales:

- El centro de datos del proveedor de servicios de IPTV.
- La red de acceso de banda ancha.
- Los dispositivos del consumidor de IPTV.

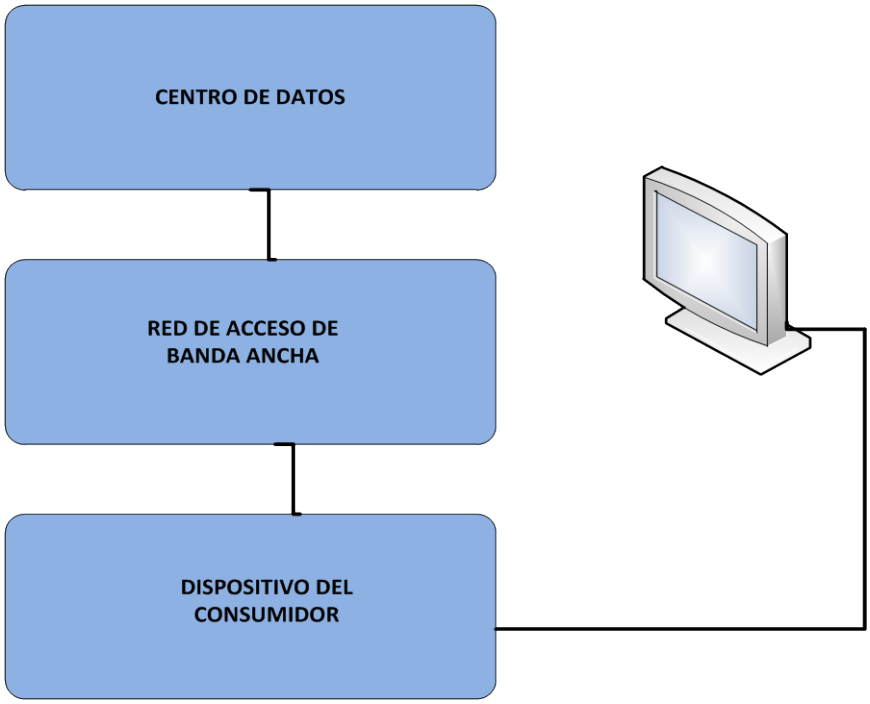

Figura 1.1 Representación en bloques de un sistema IPTV.

Centro de datos del proveedor de servicios IPTV

También conocido como “cabecera”, el centro de datos recibe el contenido de una amplia variedad de fuentes que incluye video local, productores de contenido, canales digitales terrestres, satelitales y de cable. (O’Driscoll, 2008)

Una vez recibido, el contenido de video es preparado por diferentes componentes de hardware como codificadores, servidores de video, *routers* IP y equipos dedicados a la seguridad, para ser difundido sobre el protocolo IP.

Adicionalmente este bloque requiere de un sistema de administración de suscriptores, para la configuración de perfiles de usuarios y métodos de pago.

La localización del centro de datos estará regida por la infraestructura de red con que cuente el proveedor de servicios.

Red de acceso de banda ancha

La distribución de los servicios IPTV requiere de una conexión *punto a punto*. Mientras más amplio sea el grupo de suscriptores del servicio IPTV, mayor será el número de conexiones *punto a punto* necesarias, incrementándose significativamente las demandas de

ancho de banda. Por este motivo, se hace necesario para la extensión del servicio con la calidad requerida, la utilización de redes de banda ancha, como las basadas en fibra óptica o las modernas híbridas de fibra óptica y cable coaxial. Este tema será tratado con más profundidad en epígrafes posteriores.

Dispositivos para el consumidor de IPTV

Los dispositivos para el consumidor son los componentes del sistema que permiten el acceso de las personas a los servicios de IPTV.

Estos dispositivos conectados a la red de banda ancha, son los responsables de la decodificación y procesamiento del contenido de video entrante. Soportan tecnologías avanzadas para disminuir o eliminar completamente el efecto de los problemas de la red, sobre los contenidos de IPTV. (O'Driscoll, 2008)

1.1.3 Principales servicios ofrecidos por un sistema de IPTV

Los proveedores de servicio IPTV han enfocado sus esfuerzos generalmente a dos aplicaciones fundamentales:

- La difusión de Televisión Digital.
- La distribución de contenidos bajo demanda.

Difusión de TV Digital

La difusión de televisión en formatos digitales trae consigo una serie de características que la diferencian ampliamente de los sistemas analógicos. Tales como:

- *Experiencia de visualización mejorada*: como refiere (Feng, 2001) la experiencia de visualización es mejorada a través de contenidos de alta definición, sonido de gran calidad, incremento del número de canales, la posibilidad de conmutar entre varios ángulos de cámara y una renovada forma de acceso a nuevos servicios.
- *Cobertura mejorada*: tanto la señal analógica como la digital se ven debilitadas con la distancia. Mientras que la imagen en un sistema de televisión analógica disminuye su calidad a medida que los usuarios están más alejados de las antenas de difusión, una imagen en un sistema digital se mantiene perfecta hasta que la señal se hace demasiado débil para ser recibida. (Bruin y Smits, 1999)

- *Capacidad mejorada y nueva oferta de servicios*: usando las tecnologías digitales para transmitir televisión, los proveedores de servicio pueden transportar más información que la permitida para sistemas analógicos. Los contenidos de video son comprimidos para ocupar solo un pequeño porcentaje del ancho de banda normalmente requerido por los sistemas analógicos para difundir los mismos. (Kozamernik y Vermaele, 2005)

  El ancho de banda sobrante puede ser rellenado con programas o servicios de datos como:

  - Video bajo demanda (VoD).
  - Servicios de correo electrónico e Internet.
  - Educación interactiva.
  - Comercio interactivo en televisión.

- *Flexibilidad de acceso incrementada*: tradicionalmente, solo era posible disfrutar de los contenidos de video analógico en un receptor de TV. Con la introducción de las tecnologías digitales, el video es accesible para un amplio rango de dispositivos que van desde teléfonos móviles hasta computadoras estándar. (Held, 2007)

Video bajo demanda (VoD)

Además de permitir a las compañías de telecomunicaciones difundir los canales de TV Digital tradicionales, IPTV provee acceso a un amplio rango de contenidos descargables bajo demanda. A diferencia de los servicios de televisión tradicionales donde los programas son transmitidos de acuerdo con una programación definida, VoD provee a los usuarios finales de IPTV, la facultad de seleccionar, descargar y ver contenidos a su conveniencia. Las aplicaciones que permiten el funcionamiento del servicio de VoD, generalmente incluyen bibliotecas con los títulos disponibles de los programas almacenados.

1.2 Infraestructuras de red para la distribución de servicios de IPTV

Por la naturaleza de la tecnología IPTV se necesitan rápidas plataformas de red para asegurar la difusión de los contenidos.

Una red de distribución de IPTV está compuesta por dos partes: la "última milla" de distribución de banda ancha y el núcleo o *backbone*.

1.2.1 Redes de distribución en la "última milla"

Las redes de distribución en la última milla, o redes de acceso como también se les conoce, son las más cercanas al usuario final y conectan a la cabecera o centro de datos con los dispositivos terminales del sistema. Como refieren (Kozamernik y Vermaele, 2005), existen seis tipos diferentes de redes de acceso de banda ancha, que cumplen con los requerimientos de ancho de banda demandados por IPTV:

- Redes de fibra óptica.
- Redes DSL.
- Redes HFC.
- Redes satelitales.
- Redes inalámbricas.

A continuación, se describen las características fundamentales de las redes HFC, pues constituyen el soporte sobre el cual se pretende que funcione el *set-top box* resultante de esta investigación.

Redes HFC

Una red HFC (Figura 1.2), es una red de cable que combina en su estructura el uso de la fibra óptica y el cable coaxial. Representa la evolución natural de las redes clásicas de televisión por cable (CATV). Está compuesta básicamente por una cabecera de red, la red troncal, la red de distribución, y el último tramo de acometida al hogar del abonado (acometida). (Tooley y Bowman, 2010)

La **cabecera** (*head end*), es el órgano central desde donde se gobierna todo el sistema. Suele disponer de una serie de antenas que reciben los canales de TV y radio de diferentes sistemas de distribución (satélite, microondas, etc.), así como de enlaces con otras cabeceras o estudios de televisión. Actualmente, las cabeceras han aumentado considerablemente en complejidad para satisfacer las nuevas demandas de servicios interactivos y de datos a alta velocidad. (Tooley y Bowman, 2010)

La **red troncal** es la encargada de repartir la señal compuesta generada por la cabecera a todas las zonas de distribución que abarca la red de cable.

El primer paso en la evolución de las redes clásicas todo−coaxial (CATV), hacia las redes de telecomunicaciones por cable HFC consistió en sustituir las largas cascadas de amplificadores y el cable coaxial de la red troncal por enlaces *punto a punto* de fibra óptica. La red troncal se ha convertido, en una estructura con anillos redundantes que unen nodos ópticos entre sí. En estos nodos ópticos es donde las señales descendentes (de la cabecera a usuario) pasan de óptico a eléctrico para continuar su camino hacia el hogar del abonado a través de la red de distribución de coaxial.

La **red de distribución** (Figura 1.2) está compuesta por una estructura tipo bus de coaxial que lleva las señales descendentes hasta la última derivación antes del hogar del abonado.

La **acometida** (*drops*) es la que llega a los hogares de los abonados y es sencillamente el último tramo antes de la base de conexión, en el caso de los edificios es la instalación interna.

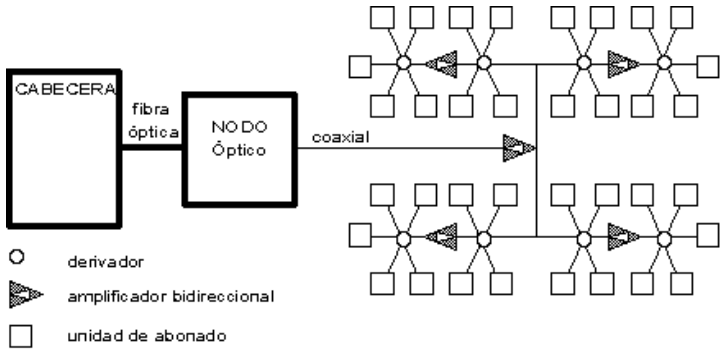

Figura 1.2 Estructura de una red HFC.

Durante los años noventa, la industria del cable desarrolló un gran número de esquemas para soportar la transmisión de datos en redes HFC, por lo que varios estándares emergieron en calidad de competidores, entre ellos, la especificación DOCSIS, utilizada en las redes de cable cubanas. Este tema será abordado con más profundidad en el siguiente epígrafe, pues las particularidades de DOCSIS son precisamente, las que hacen que una red HFC pueda ser utilizada como soporte para IPTV.

1.2.2 Backbone o núcleo

El *backbone* o núcleo de una infraestructura de red IPTV, es el encargado de transportar una gran cantidad de video a una velocidad alta, entre el centro de datos del proveedor de servicios IPTV hasta la red de distribución de última milla (Figura 1.3). Tres de las infraestructuras de red más utilizadas como *backbone* son las redes ATM sobre SONET/SDH, las redes IP sobre MPLS y las redes Metro Ethernet. Debido al alcance de la investigación, estos tipos de redes no serán abordados con profundidad.

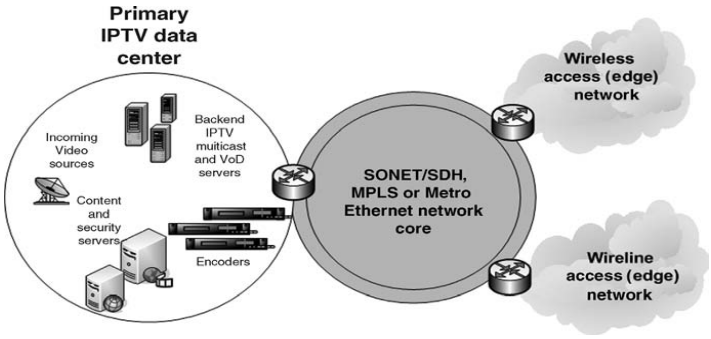

Figura 1.3 Redes del núcleo o backbone de un sistema IPTV. (O'Driscoll, 2008)

1.3 Especificación DOCSIS

DOCSIS® es una marca registrada de CableLabs® (Cable Television Laboratories, Inc.), consorcio de investigación y desarrollo sin ánimos de lucro integrado por *Arris*, *BigBand*, *Broadcom*, *Cisco Systems*, *Harmonic*, *Intel*, *Motorola* y *Texas Instruments.*

Es además, un estándar no comercial que dicta los requerimientos de una interfaz de comunicaciones, para soportar el tráfico de datos IP en sistemas de cable. Se describe en un grupo de especificaciones y reportes técnicos que exponen cómo deben operar los dispositivos y sus interfaces en capa Física (PHY), Control de Acceso al Medio (MAC) e Internet. (CableLabs, 2006)

La versión europea de DOCSIS se denomina EuroDOCSIS, y la principal diferencia es que en Europa los canales de cable tienen un ancho de banda de 8 MHz, mientras que en Norte América es de 6 MHz. La primera versión de la especificación (DOCSIS 1.0) surgió en Marzo de 1997, y tuvo una gran revisión (DOCSIS 1.1) en Abril de 1999, que fundamentalmente adicionó algún soporte de calidad de servicio (QoS - *Quality of Service*). La versión 2.0 (DOCSIS 2.0) se lanzó en Diciembre de 2001, con una mejora en las

velocidades de subida. Posteriormente en Agosto de 2006, aparece DOCSIS 3.0, para proveer mayores velocidades tanto en subida como en bajada, así como dar soporte a IPv.6. El apoyo de la industria a DOCSIS ha hecho del despliegue de las redes HFC un líder en servicios de banda ancha en el principal mercado del mundo, el estadounidense, manteniendo a su favor el 50% de las acciones del sector.

1.3.1 Equipamiento DOCSIS

Implementar DOCSIS implica la puesta en funcionamiento de un equipo denominado Sistema de Terminación de Cable Módems o CMTS (*Cable Modem Termination System*) en la cabecera, que controla los puertos de envío y recepción de datos hacia la red de cable. Asimismo, en el extremo del cliente debe colocarse un dispositivo que se conoce como Módem de Cable o CM (*Cable Modem*).

El CMTS básicamente conecta Internet y la red interna del proveedor (*Back Office Network*) con la red de cable o HFC, reenviando paquetes entre esos dos grandes dominios y entre los canales de subida y bajada dentro la propia red HFC.(Cable Television Laboratories, 2010)

Para dar soporte a la red DOCSIS, se emplean además algunas aplicaciones, cuyo conjunto se conoce como Sistemas de Aprovisionamiento (*Provisioning Systems*):

- **Servidores DHCP** (*Dynamic Host Configuration Protocol*): proveen información de configuración inicial como la dirección IP de los CMs y otras direcciones.
- **Servidor de Archivos de Configuración** (TFTP *Trivial File Transfer Protocol*): permite que los CMs descarguen sus archivos de configuración cuando arrancan.
- **Servidor de Descarga de Software**: permite actualizar el software de los fabricantes en el equipamiento.
- **Servidor de Protocolo de Tiempo** (*Time Protocol Server*): proporciona la referencia de tiempo para la red.
- **Servidor de Revocación de Certificados** (*Certificate Revocation Server*): provee estados de certificados.

De manera similar, el Sistema de Gestión de la Red (NMS *Network Management System*):

- **Gestor SNMP** (*Simple Network Management Protocol Manager*): permite al operador configurar y monitorear el equipamiento, típicamente los CMs y el CMTS.
- **Servidor Syslog**: recolecta mensajes acerca de la operación de los dispositivos.
- **Servidor de IPDR** (*Internet Protocol Detail Record*): permite al operador obtener estadísticas de manera general y eficiente respecto al uso de la transmisión de datos.

DOCSIS funciona principalmente entre las capas 1 y 2 del modelo OSI (*Open Systems Interconnection Reference Model*) de la ISO (*International Organization for Standarization*), conocidas también como Capa Física (PHY) y Capa de Control de Acceso al Medio (MAC). En términos simples, las redes DOCSIS son redes IP sobre HFC, en la Figura 1.4 puede apreciarse la estructura de un sistema IPTV utilizando esta tecnología.

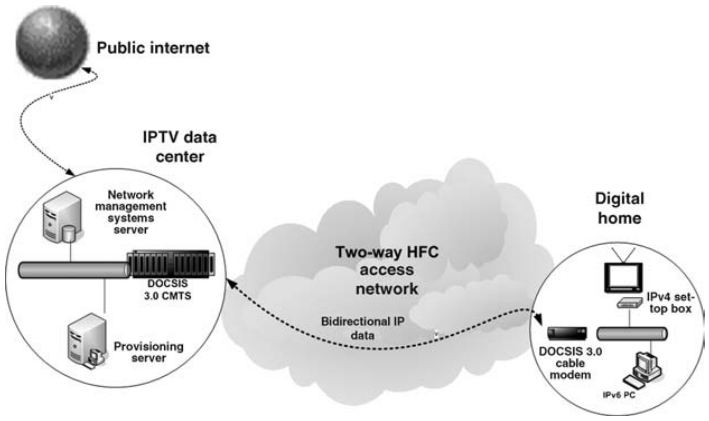

Figura 1.4 IPTV utilizando la especificación DOCSIS 3.0. (O'Driscoll, 2008)

1.3.2 Características generales de DOCSIS

El espectro total es dividido en una parte para la banda de subida o retorno desde 5 MHz hasta 42 MHz (en algunos sistemas hasta 85 MHz, conocida como banda extendida de *upstream*), y la parte para la bajada a partir de los 108 MHz hasta la capacidad de la red. Así DOCSIS especifica un camino asimétrico de datos con flujos de bajada (*downstream*) y subida (*upstream*) en dos frecuencias separadas. Las portadoras de bajada y subida proveen dos canales compartidos para todos los CMs.

Downstream

La banda de *downstream* se comparte con los demás servicios que brinda la red. El ancho de banda para cada canal descendente puede ser de 6 MHz ó de 8 MHz para cumplir con las normas para los canales de difusión de televisión en Norteamérica y Europa. Los formatos

de modulación que pueden ser empleados son: 64-QAM y 256-QAM. (Cable Television Laboratories, 2010)

Los paquetes enviados por el canal de bajada se separan en tramas MPEG de 188 bytes, con 4 bytes de encabezado y 184 bytes de carga útil. Aún cuando a todos los CMs llegan todas las tramas, normalmente se configuran para recibir solamente las que están direccionadas a su dirección MAC o a la dirección de difusión.

En el sentido descendente, DOCSIS emplea un sistema *primero-que-llega primero-que-se-atiende* para el reenvío de paquetes, así los paquetes que arriban de Internet se reenvían inmediatamente como paquetes DOCSIS y se modulan en un canal *downstream* de RF para la transmisión. (Tooley y Bowman, 2010)

Upstream

Las características de esta banda (ruido y distorsión) hacen que se necesiten mecanismos robustos, como modulaciones eficientes, para la transmisión de datos. Por tal motivo se conforman también los CMTS con más puertos de subida que de bajada, permitiendo segmentar más la red en el sentido ascendente y lograr mejores niveles de señal.

En la banda de subida, DOCSIS emplea dos esquemas básicos de acceso al medio: modo FDMA/TDMA (*Frequency Division Multiple Access/Time Division Multiple Access* - Acceso Múltiple por División de Frecuencias/Acceso Múltiple por División de Tiempo), descrito como TDMA, y modo FDMA/TDMA/S-CDMA descrito como S-CDMA (*Synchronous Code Division Multiple Access* o Acceso Múltiple por División de Código Sincrónico), permitiendo seis tasas de modulación y múltiples formatos para esta.

Con FDMA se divide la banda en múltiples canales de frecuencias. TDMA responde a la naturaleza de ráfaga de las transmisiones de subida y permite compartir un canal de RF de subida entre varios cable módems a través de la asignación dinámica de ranuras de tiempo. S-CDMA da la posibilidad de que varios CMs transmitan simultáneamente en el mismo canal de RF y sobre la misma ranura de tiempo TDMA, separados por diferentes códigos ortogonales. (Cable Television Laboratories, 2010)

Los tipos de modulación establecidos para los moduladores/demoduladores de subida son: QPSK (*Quadrature Phase Shift Keying* o Desplazamiento de Fase en Cuadratura), 8-QAM

(*Quadrature Amplitude Modulation* o Modulación por Amplitud en Cuadratura), 16-QAM, 32-QAM, 64-QAM y 128-QAM TCM (*Trellis Code Modulation* o Modulación con Codificación de Trellis).

El CMTS reserva el ancho de banda de subida basado en peticiones de los CMs y las políticas de calidad de servicio (QoS - *Quality of Service*). El canal de subida se divide en ranuras de tiempo multiplexadas, llamadas "mini-slots", de una duración de 6.25 µs, y la oportunidad para transmitir en ellas es administrada por el CMTS, que las identifica mediante referencias de tiempo. La influencia de la distancia en el retardo implica una compensación de tiempo, *timing offset*, en las referencias de los CMs para que identifiquen con suficiente precisión la posición de los mini-slots. (Cable Television Laboratories, 2010)

1.3.3 Funcionamiento básico

Básicamente, para el funcionamiento de la red DOCSIS se debe tener en funcionamiento el CMTS, un servidor DHCP y uno TFTP, de modo que los CMs puedan registrarse. Normalmente se emplea un DHCP de dos niveles o rangos, uno para la operación de los CMs, o sea, a nivel de servicio, y otro para los equipos de premisas del cliente o CPEs (*Customer Premises Equipment*).

Cuando un cable módem se conecta a la red coaxial, comienza a rastrear un canal descendente buscando en toda la banda una transmisión 64-QAM ó 256-QAM, que contenga información válida acerca de un canal de subida, para poder establecer una comunicación bidireccional. Si efectivamente lo encuentra y lo retiene, comienza lo que se conoce como *Ranging*. El *Ranging* es un proceso que consta de dos etapas, la primera es el *Ranging* Inicial, en la que se configura el menor nivel de potencia que garantiza la comunicación. Después pasa a un nuevo estado conocido como *Station Maintenance Ranging*, en el que el cable módem recibirá instrucciones desde el CMTS para hacer ajustes en cuanto a frecuencias de transmisión, amplitud, *timing offset* y pre-ecualización. El *Station Maintenance Ranging* ocurrirá al menos una vez cada 30 segundos para cada cable módem, realizando ajustes de forma continua y para que el CMTS conozca que los módems se encuentran en línea.

De esta manera, el CMTS periódicamente envía tramas de gestión y otra clase de mensajes denominados mensajes MAP (*Media Access Protocol* o Protocolo de Acceso al Medio)

para que los CMs puedan identificar futuras ranuras TDMA para la subida (*upstream*). Inmediatamente al contar con ancho de banda para transmitir, el CM obtiene su IP y otras direcciones importantes mediante DHCP y descarga el archivo de configuración del servidor TFTP, con los parámetros que necesita para el acceso a la red: velocidades, QoS y otras configuraciones. Si las pruebas de validez son satisfactorias, se concluye el proceso con el Registro del CM en la red y se le asigna un Identificador de Servicio o SID (*Service IDentifier*). Finalmente se permite a los suscriptores transmitir tráfico IP. (Downey, 2006)

1.3.4 DOCSIS 3.0

La gran evolución del estándar es DOCSIS 3.0, una serie de especificaciones que define la tercera generación de la transmisión de datos de alta velocidad sobre sistemas de cable. La flexibilidad en la implementación de sus nuevas características y la escalabilidad de los equipos de terminación lo hacen muy atractivo.(Vecima Networks, 2008)

La principal característica que aporta la versión 3.0 de DOCSIS es el *Channel Bonding* (Puenteo de Canales), que permite aumentar el ancho de banda, tanto en subida como en bajada, para cada suscriptor. Además cuenta con: (Pularikkal, 2009)

- Mayores rangos de direccionamiento IP con IPv6 (*Internet Protocol version 6*)
- Soporte mejorado para *Multicast* (multidifusión)
- Técnicas de encriptación más robustas con AES (*Advanced Encryption System*) y MMH (*Multilinear Modular Hashing*)
- Sistemas de gestión de más capacidad con mediciones más profundas.
- Razón canales de subida-canales de bajada más flexible (U:D) (*Upstream Downstream Ratio*).

Las especificaciones DOCSIS 3.0 no limitan la cantidad de canales que se pueden unir para el *channel bonding*. En la práctica los factores que lo limitan pueden describirse como: cantidad de canales de 6 MHz u 8 MHz libres y equipamiento que soporte mayores grupos de canales puenteados. Pocos fabricantes pueden ofrecer soluciones económicas de equipamiento que puedan puentear más de 4 canales debido al alto costo marginal y a la baja demanda. (Tooley y Bowman, 2010)

1.4 Streaming de contenidos de IPTV sobre la red

Los patrones de tráfico en tiempo real de IPTV difieren de los patrones generados por otros servicios basados en el protocolo IP, tales como VoIP y el acceso rápido a Internet. El tráfico de video, demanda altos niveles de rendimiento, mientras que el tráfico de Internet fluctúa entre niveles altos y bajos de actividad. Es por ello que se necesita un tratamiento especial de los contenidos de video para poder garantizar los niveles adecuados de calidad de servicio. Dicho tratamiento especial es proporcionado por las técnicas de *streaming*.

¿Qué significa la palabra '*streaming*'? '*Stream*' significa 'chorro' o 'flujo' y alude a la descarga de un fichero que no tiene principio ni final: el caso más claro es una transmisión en directo. (López y Badia, 2008)

Se entiende por *streaming* la capacidad de distribución de contenido multimedia, con la característica de poder visualizar estos contenidos en el cliente mientras la información está siendo trasmitida por la red. No es necesario descargar completamente el contenido multimedia para comenzar su visualización, este se va almacenando en un buffer y se va ejecutando al mismo tiempo que se desarrolla la transmisión. (Sciara, 2004)

En este epígrafe se describen las características de los tres tipos básicos de *streaming* existentes: Unicast, Multicast y Broadcast. Se analizan además algunos de los protocolos más usados para el streaming de video sobre redes IP.

1.4.1 Unicast

Unicast entrega *streams* uno a uno para cada cliente y se utiliza para el Video on Demand (VoD), ya que cualquier usuario puede hacer una petición de una secuencia de *stream* en cualquier momento. Es similar, a tener un videoclub abierto las 24 horas, pero sin la obligación de ir a por la película ni de bajar a devolverla. (Ortiz y Serna, 2006)

El servicio Unicast consiste en un servidor, que envía paquetes de datos a cada computador que solicita un *stream*. El método de entrega Unicast es una buena opción para recibir transmisiones en tiempo real, pero tiene sus desventajas. El servidor, debe enviar el flujo de datos individualmente a todo aquel, que quiera recibir la transmisión (Figura 1.5). Si se tiene a unas cuantas personas recibiendo el *stream,* el funcionamiento es correcto, pero si se

trata de difundir el material a miles de usuarios se deberá considerar dos inconvenientes con el proceso de Unicast: demasiadas peticiones y demasiados paquetes.

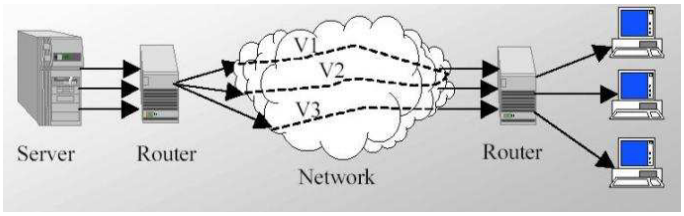

Figura 1.5 Video Streaming Unicast.

Con Unicast, el servidor tiene que procesar cada solicitud de *streams* y atenderla. Cada *stream*, utiliza una pequeña porción de procesamiento del servidor. Si el servidor posee muchas solicitudes no podrá sostener la sobrecarga y muchas personas no podrán recibir la transmisión. Este es el mismo problema que tienen los servidores de archivos FTP si se intenta descargar un archivo, que todo el mundo quiere en el mismo momento.

El problema existente con Unicast, y un gran número de solicitantes simultáneos de *stream*, es que una serie separada de paquetes de datos debe ser enviada a cada ordenador. Incluso, si el servidor pudiera hacer esta tarea, el número de paquetes de datos en tránsito por la red inundaría el sistema entero haciendo que la transmisión se volviera muy lenta, o incluso se detuviese. (Apostolopoulos et al., 2002)

1.4.2 Multicast

Al contrario que el Unicast, Multicast entrega los *streams* simultáneamente, del servidor a muchos clientes. Es conocido como *Near Video on Demand* (NVoD) cuando actúa sobre subredes, ya que los usuarios deben ver el mismo contenido a la misma vez. Es similar, a un *pay-per-view* de la Televisión por cable, donde un grupo de usuarios están autorizados a ver el programa, que se emite. Ofrece la difusión a varios clientes de un mismo contenido (Figura 1.6), ya sea creado en ese momento en vivo (*live broadcast*), o almacenado previamente en el servidor. El usuario no posee el control, como si fuera un video doméstico, sin poder realizar las acciones propias del mismo (posicionamiento, paro, retroceso o avance rápido, etc.), pero sí puede parar y volverse a conectar a la emisión.

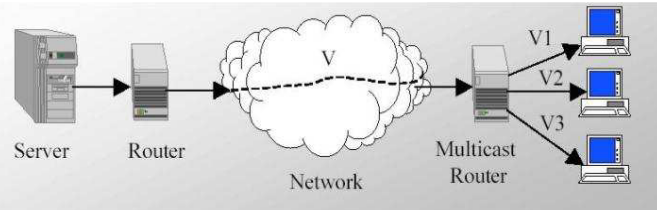

Figura 1.6 Video Streaming Multicast

Multicast hace su trabajo de transmisión de manera similar a cómo funcionan los canales de televisión o las estaciones de radio: el programa (archivo de audio/video) se emite desde la estación hacia los transmisores (servidores *streaming* conectados a la red), quienes se encargan de distribuir la señal (el *stream*) a los televidentes. Cuando el espectro de televidentes (usuarios, visitantes) se extiende, se agregan repetidores (servidores, lo que es conocido como *splitting*).

Debido a que Multicast, utiliza el protocolo UDP, para la transmisión de una serie de paquetes de datos, no existe una manera sencilla de que el reproductor solicite un paquete de datos para que sea enviado de nuevo. Esto quiere decir que algunos paquetes se pierden, incluso antes de que el usuario pueda notarlo, debido en parte, a la manera en que el reproductor codifica los archivos y los enumera. (Apostolopoulos et al., 2002)

Si se pierden una gran cantidad continua de paquetes será perceptible por el usuario en forma de cortes de sonido y de pausado de imágenes, siempre dependiendo de la amplitud del *buffer* que se haya configurado por el usuario.

1.4.3 Broadcast

Broadcast es un caso especial de Multicast, en el que se entrega un único *stream*, simultáneamente, a toda la red. Este método es utilizado, por ejemplo, para la retransmisión en directo de la presentación ejecutiva en una empresa. Todos los usuarios pueden ver en la red corporativa la difusión en una hora acordada.

1.4.4 Protocolos

Existen protocolos creados específicamente para la transmisión de video sobre la red. Algunos de los más utilizados son:

Protocolo RTP (Real Time Protocol): es el protocolo más importante normalizado para *streaming*. Todos los *streams* de contenidos, sin tener en cuenta su formato y contenido, son encapsulados en paquetes RTP. (Ortiz and Serna, 2006)

RTP dispone de varios campos para datos que no se encuentran presentes en TCP, en particular un *timestamp* y un número de secuencia, funciona sobre UDP y utiliza sus aplicaciones de multiplexado y *checksums*, aunque puede soportar otros protocolos de transporte. Permite el control del servidor de *streaming* para que los *streams* de vídeo sean servidos a la velocidad adecuada. El reproductor está, entonces, en condiciones de poder organizar los paquetes RTP recibidos en el orden correcto y reproducirlos a la velocidad adecuada. Transmite paquetes en tiempo real tanto de audio como de video. No garantiza por sí sólo la entrega de los datos en tiempo real, pero provee de los mecanismos de envío y recepción necesarios a las aplicaciones, para que puedan soportar los datos *streaming*. Ofrece entrega de datos Multicast siempre que la red subyacente soporte este servicio. (Sciara, 2004)

Los paquetes perdidos o dañados no son retransmitidos. Existen herramientas de software que le permiten al cliente subsanar problemas originados por la pérdida de bits (replicación de paquetes, canceladores de error, etc…). Si la velocidad de conexión es inferior a la velocidad de transferencia de datos necesaria, la transmisión se corta y la calidad de la reproducción se degrada o no se completa. Si la velocidad de la conexión es rápida, el ancho de banda excedente permanece sin utilizar y la carga del servidor solamente depende del ancho de banda del *stream*.

Protocolo RTSP (Real Time Streaming Protocol): es un estándar desarrollado por la Universidad de Columbia en el RFC 2326 para el IETF. El RTSP es un protocolo cliente-servidor de nivel de aplicación para el control del flujo de datos en tiempo real. Es básicamente un protocolo de señalización o de control, pero también provee de algunas funcionalidades de calidad de servicio. Utiliza el RTP como protocolo subyacente para la entrega de datos y ofrece al usuario los controles: Play, Stop, Pause, FF, REW; así como, acceso aleatorio a cualquier corte del *stream*. Facilita al servidor el ajuste de la velocidad de transmisión de los datos al ancho de banda de la conexión ante congestiones en la red, con el fin de utilizar lo mejor posible la capacidad disponible. (Ortiz y Serna, 2006)

Una vez establecido, el RTSP puede controlar uno o varios *streams* sincronizados temporalmente, como el audio y el video. Las fuentes de datos pueden incluir tanto datos fijos como en tiempo real. Proporciona además mecanismos para seleccionar canales de envío (como el UDP, UDP Multicast, TCP) y mecanismos de entrega continúa de *streams* basados en RTP. Se puede considerar que el RTSP actúa "como control remoto de la red" para los servidores multimedia. (Sciara, 2004)

Otra importante función del RTSP es la de elegir el canal óptimo de conexión para el cliente con los mecanismos proporcionados. Por ejemplo, si el UDP no puede utilizarse (algunos cortafuegos no lo permiten), el servidor de streaming ofrece la elección de protocolo: UDP multidifusión o TCP para clientes específicos. Al poder seleccionar el protocolo subyacente, se puede utilizar tanto para difusión (Unicast), como multidifusión (Multicast). (Apostolopoulos et al., 2002)

RTCP (Real Time Control Protocol): es usado conjuntamente con el RTP y utiliza TCP para la conexión bidireccional cliente-servidor. Es un protocolo que da calidad de servicio al sistema y proporciona mecanismos de realimentación para informar de la calidad en la distribución de los datos, que implican la transmisión periódica de paquetes de control a todos los participantes de una sesión. (Ortiz y Serna, 2006)

Los mensajes, que se reciben, incluyen informes relativos al número de paquetes perdidos y estadísticas de *jitter*. Esta información puede ser utilizada por aplicaciones de nivel superior para el control de la sesión y mejora de la transmisión, por ejemplo, la razón de bits de un *stream* puede modificarse para combatir la congestión de red, se puede diagnosticar fallos en la distribución o se puede usar para controlar un mecanismo adaptativo de codificación, que responda a las condiciones de la red. (López y Badia, 2008)

Los paquetes de RTCP se envían de modo que el tráfico en la red no aumenta linealmente con el número de agentes participantes en la sesión, es decir el intervalo de envío se ajusta de acuerdo al tráfico.

1.5 Dispositivos del consumidor de IPTV

Los servicios de IPTV pueden ser accedidos a través de dispositivos que se encuentran en el lado del cliente, denominados *set-top boxes*.

Un *set-top box*, es una computadora de propósito específico que traduce las señales obtenidas de cualquiera de los soportes de red analizados anteriormente, a un formato que puede ser visto en una pantalla de televisión. Son distribuidos generalmente por los proveedores de servicios.

Para IPTV se utilizan los *set-top boxes* basados en el protocolo IP (*IP set-top box*). En la Figura 1.7 puede apreciarse un dispositivo de este tipo disponible en el mercado.

Figura 1.7 Set-top box para IPTV distribuido por la firma Philips.

1.5.1 Características generales y clasificación

Algunas de las características generales de los set-top boxes utilizados para IPTV, definidas por (O'Driscoll, 2008), son las siguientes:

- ***Presentación de señales de audio y video sincronizadas en una pantalla de televisión***: similar al set-top box de cable, el set-top box IP toma el contenido de audio y video digital interactivo y personalizado, lo decodifica, y lo muestra en un televisor.
- ***Procesamiento lineal de señales Multicast de televisión***: los set-top boxes IP son capaces de acceder a aplicaciones de Multicast de TV.
- ***Manipulación de la degradación de servicio***: Las redes de acceso de banda ancha no están exentas de problemas tan comunes como la pérdida de tramas y la congestión, sin embargo los set-top boxes IP son capaces de compensar estos efectos, sin que se vea afectado el servicio en general.
- ***Interfaz de usuario personalizable***: Muchos *set-top boxes* incluyen software que permiten a los proveedores de servicios agregar su propio sentido a la interfaz de usuario de Televisión.

Los *set-top boxes* para IPTV, pueden clasificarse según las prestaciones incorporadas que poseen, de la siguiente forma:

- ***Unicast y Multicast***: son los que por sus características de hardware pueden soportar ambas funcionalidades, Unicast y Multicast. (Apostolopoulos et al., 2002)
- ***DVRs (Digital Video Recorders)***: estos dispositivos permiten a los suscriptores, la reproducción y rebobinado de programas en vivo, la grabación de películas y shows en el disco duro que traen integrado en su interior, la reproducción de los contenidos de video en cualquier momento y la distribución del contenido de video almacenado para otros equipos del hogar. (Bruin y Smits, 1999)
- ***Híbridos***: estos dispositivos son capaces de procesar tanto los contenidos de IPTV como los contenidos de la Televisión Digital Terrestre, Satelital o por Cable. Por tanto, con equipos de este tipo los suscriptores pueden disfrutar de los contenidos de TV transmitidos por las vías digitales convencionales y además de un servicio de video bajo demanda brindado por la tecnología IPTV. A estos dispositivos se les ha incorporado además, la tecnología para grabación digital. (Bruin y Smits, 1999)

1.5.2 Componentes de software de un set-top box

Se denominan “componentes de software”, al grupo de elementos de software que hacen posible que un *set-top box* brinde todas las prestaciones anteriormente tratadas. Existe una amplia variedad de dichos elementos, los cuales se han agrupado por categorías (Figura 1.8) para su mejor descripción: (O’Driscoll, 2008)

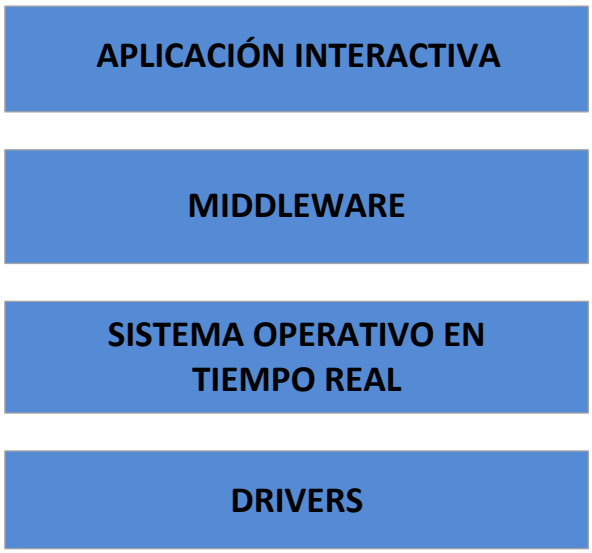

Figura 1.8 Componentes de software de un set-top box.

- ***Drivers***: dentro de esta categoría se encuentran los componentes de software necesarios para el control y acceso a los elementos de hardware.
- ***Sistema Operativo en Tiempo Real (RTOS- Real Time Operative System)***: este sistema provee una variedad de funcionalidades para la realización del análisis y procesamiento de los streams entrantes y tareas generales de configuración.
- ***Middleware***: este tipo de software constituye un puente de comunicación entre el sistema operativo y la aplicación interactiva con la que interactúa el usuario en los sistemas de IPTV.
- ***Aplicaciones interactivas****:* estas aplicaciones son las encargadas de mostrar los contenidos y servicios al usuario, a través de interfaces gráficas atractivas.

1.5.3 Conclusiones del capítulo

En este capítulo se han descrito los fundamentos teóricos necesarios para comprender el funcionamiento de un sistema general de IPTV. Se han tratado las infraestructuras de red más utilizadas para la distribución de los servicios de esta tecnología, enfatizando en las redes HFC y la especificación DOCSIS, debido su importancia para el presente trabajo. Además se han analizado, las características fundamentales del *video streaming* y los principales protocolos utilizados para tal fin. Por último, se han detallado las principales características de un set-top box para IPTV, así como la estructura de sus componentes de

software, con el objetivo de familiarizarnos con el dispositivo que pretendemos sustituir con el resultado de la presente investigación.

CAPÍTULO 2. MATERIALES Y MÉTODOS

En este capítulo se describen los principales módulos que se utilizaron para crear un *set-top box* para IPTV completamente funcional utilizando software libre. Los diferentes epígrafes están organizados de manera tal que den un seguimiento lógico a la conformación del equipo final.

2.1 ¿Por qué implementar un set-top box personalizado?

Debido a la necesidad de la empresa TeleCable Internacional de ampliar su gama de servicios, se optó por utilizar IPTV. Como se analizó en el capítulo anterior, el equipamiento DOCSIS permite la transmisión de datos sobre una red de cable, convirtiéndola así en un soporte óptimo para servicios característicos de esta tecnología, tales como: la difusión de contenidos multimedia bajo demanda y la difusión de canales de TV en vivo. Después de analizar las ofertas comerciales de los principales suministradores se concluyó lo siguiente:

- Las estructuras cliente-servidor para brindar servicios de IPTV, disponibles en el mercado, son demasiado costosas en correspondencia con la situación y prioridades económicas de Cuba.
- Los sistemas disponibles no satisfacen las necesidades de personalización de los servicios, pues imponen una limitante de diseño y configuración determinada por el fabricante.

Para franquear estos obstáculos, se ha decidido implementar un *set-top box* personalizado en correspondencia con las verdaderas necesidades que tiene el país, que pueda ser modificado a voluntad de las empresas que lo utilicen.

2.2 Herramientas de código abierto

El set-top box estará formado por dos componentes fundamentales: un componente de software y un componente de hardware (Figura 2.1). El componente de software deberá tener su mayor fuerza en la reproducción de contenidos multimedia, debido a que los servicios más básicos de la tecnología IPTV así lo requieren. Además, deberá funcionar en cualquier hardware de regulares prestaciones, asegurando así su portabilidad y facilitando la conformación del producto final. Por tanto, los esfuerzos estarán centrados en encontrar el software óptimo que reúna estas características. En este epígrafe se analizan las aplicaciones disponibles de código abierto, más cercanas a la herramienta buscada.

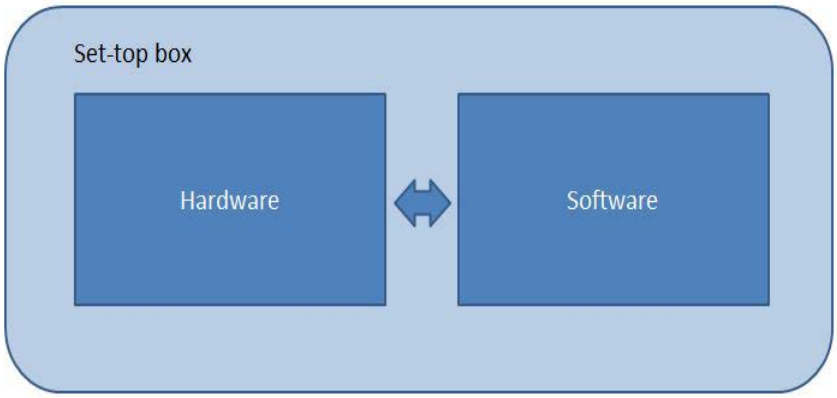

Figura 2.1 Estructura general del set-top.

2.2.1 Elisa

Elisa es un centro multimedia escrito en el lenguaje de *scripting* Python. Es multiplataforma, puede ejecutarse en Linux, Mac OS X y Microsoft Windows. No es capaz de manejar las señales de TV, pero reproduce datos provenientes de Internet o de un disco local y se ejecuta bastante rápido. Para controlar a Elisa se puede utilizar el teclado, el ratón o un mando a distancia preconfigurado. Puede reproducir videos y música desde servidores compatibles *UPnP* o *DAAP* automáticamente. El gestor de ficheros de vídeo es claro y atractivo. El aspecto visual de Elisa es elegante (Figura 2.2). Gracias al consistente uso de OpenGL, la navegación es rápida incluso en máquinas poco modernas como un Athlon XP 2500+. (Read y Schafer, 2009)

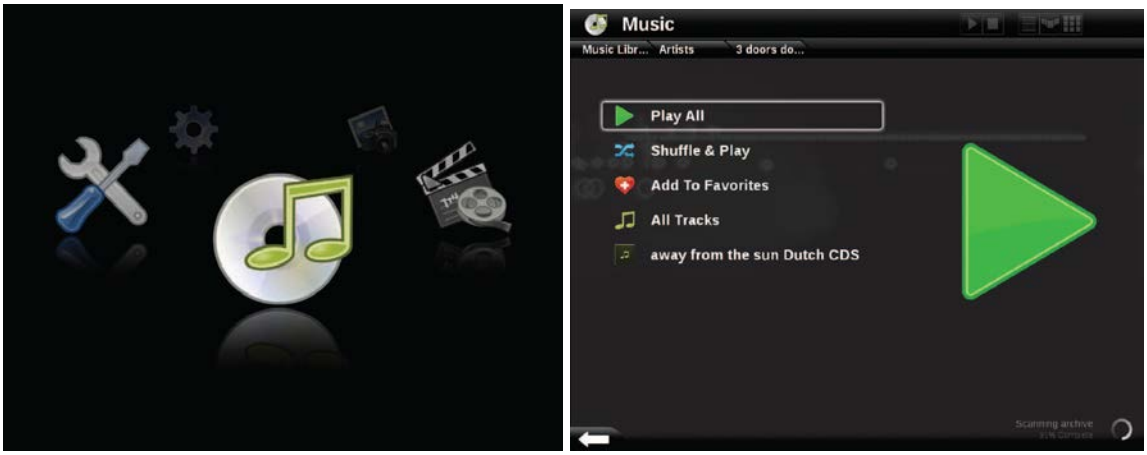

Figura 2.2 Interfaz gráfica de Elisa.

2.2.2 Freevo

Freevo es un juego de palabras en inglés, suma de “Free”, libre, y “TiVo”. TiVo, en EEUU y en otros países, es un aparato de vídeo muy popular que permite a los usuarios ver las retransmisiones en otro momento (*time shift*), además posee una guía electrónica de programas (EPG) para el contenido televisivo. Como Elisa, Freevo está programado en Python. Ofrece la experiencia televisiva con función de pausa, grabación mientras se ve y borrado automático de los anuncios. El sistema completo arranca rápidamente y aparece con una interfaz muy ordenada (Figura 2.3). Posee también fuentes fácilmente legibles y atractivas. La estructura simple de sus menús es adecuada tanto para televisores digitales como para los analógicos.

Antes de utilizarse por primera vez debe configurarse el sistema, mediante lo que podría denominarse un sistema de configuración clásico, pues las herramientas gráficas no están disponibles para este fin. Los cambios hay que teclearlos directamente en un fichero Python, *local_conf.py*. (Read y Schafer, 2009)

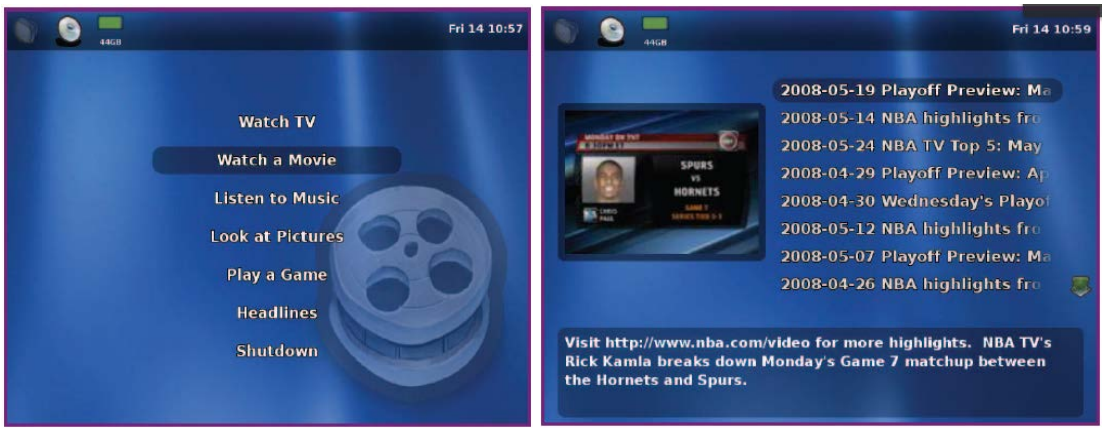

Figura 2.3 Interfaz gráfica de Freevo.

2.2.3 XBMC

XBMC (también conocido como "*XBox Media Center*") es un Centro Multimedia de entretenimiento multiplataforma bajo la licencia GNU/GPL. Inicialmente fue creado para la primera generación de la consola de juegos Xbox. El equipo de desarrollo detrás de XBMC ha portado el producto para que pueda correr de manera nativa en Linux, Mac OS X (Leopard, Tiger y Apple TV) y los sistemas operativos de Microsoft Windows. Presenta además, la ventaja de contar con una versión en Live CD denominada "XBMC Live", instalable de forma completa en una unidad flash USB o en un disco duro, ya que posee un sistema operativo embebido basado en Linux. (XBMCTeam, 2012)

XBMC soporta una amplia gama de formatos multimedia prácticamente de cualquier fuente, incluidos Internet y LAN *shares* y presenta una interfaz gráfica elegante y fácil de utilizar (Figura 2.4).

A través de su sistema de plug-in basado en Python, XBMC es expansible gracias a *add-ons* que incluyen características como guías de programas de televisión, YouTube, soporte a adelantos en línea de películas. XBMC también funciona como una plataforma de juegos; permite a usuarios jugar mini-juegos basados en Python sobre cualquier sistema operativo.

XBMC en su conjunto se distribuye bajo la licencia GNU *General Public License* (con algunas librerías utilizadas por XBMC bajo la licencia LGPL). Es un proyecto *hobby* que sólo es desarrollado por voluntarios en su tiempo libre. No es producido, aprobado, o respaldado por Microsoft u otro vendedor.

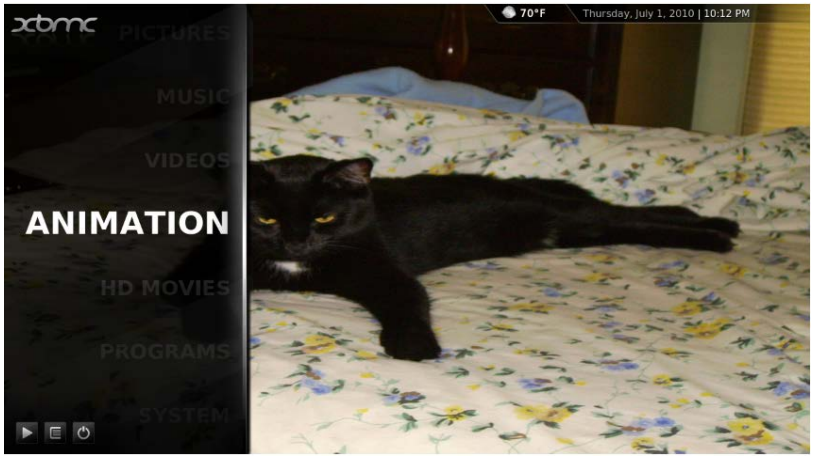

Figura 2.4 Interfaz gráfica del XBMC.

2.3 Selección y estudio de la herramienta óptima

Luego de analizar las opciones disponibles, se seleccionó la aplicación XBMC por las siguientes razones:

- Es un software que se distribuye con un sistema operativo *Open Source* subyacente, lo cual contribuye a su funcionamiento independiente en cualquier hardware de características regulares.
- Contiene un reproductor multimedia completo y potente.
- Cuenta con una interfaz gráfica elegante y es completamente *skinneable*..
- Puede ser extendido mediante *plugins* escritos en Python, uno de los lenguajes de programación más utilizados en la actualidad.

En este epígrafe se tratan las características generales del XBMC, así como las interfaces a través de las cuales, es posible interactuar con él para su modificación.

2.3.1 Estructura del software XBMC

El software XBMC está compuesto por varios bloques funcionales (Figura 2.5), donde se encuentran agrupadas todas sus prestaciones.

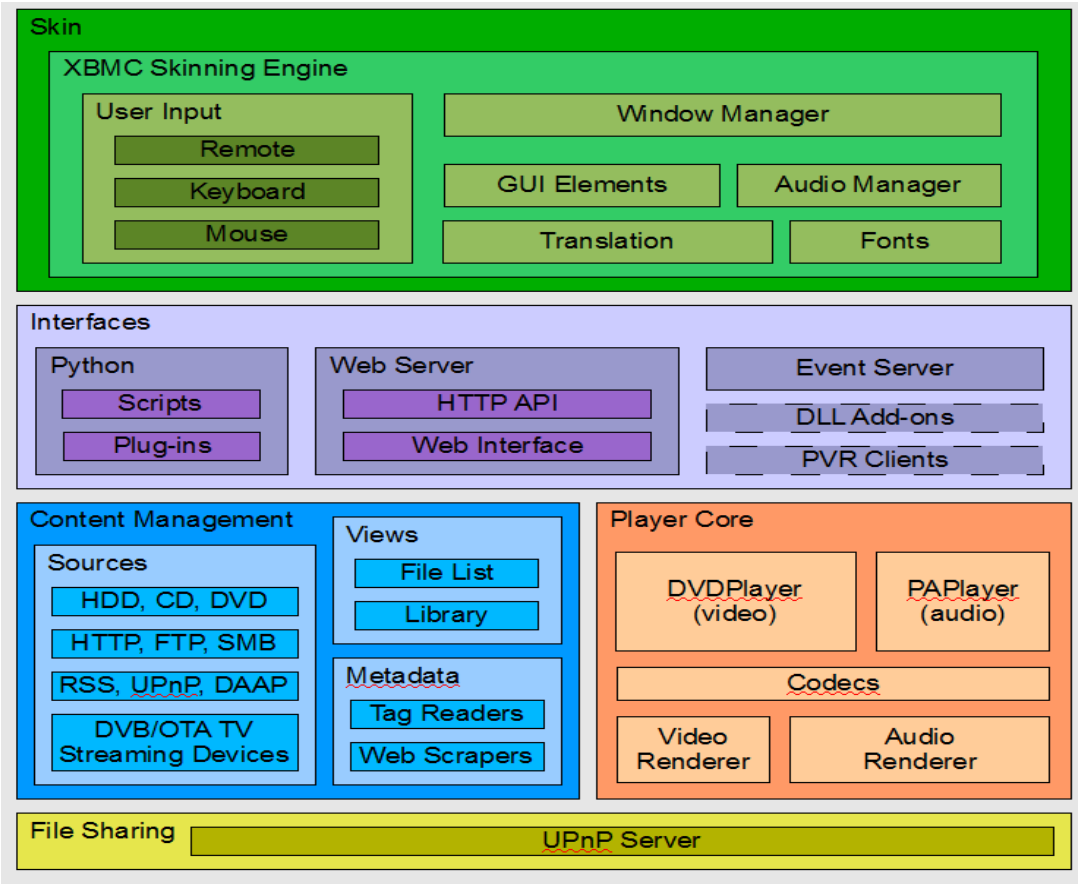

Figura 2.5 Estructura del software XBMC. (Nelson, 2009)

Interfaz Gráfica de Usuario y Procesador de *Skinning*

El núcleo (XBMC core) define un grupo mínimo de controles por cada ventana, para realizar las funciones básicas de la aplicación. Sin embargo, es posible interactuar con la interfaz gráfica de usuario (Graphical User Interface - GUI), creando *skins* o pieles (skinning) propias, donde no solo se presenten los controles básicos, sino tantos como se deseen.

Los desarrolladores de *skins* para XBMC, tienen todos los elementos gráficos usuales a su disposición; *labels, buttons, radio buttons, text boxes, spin boxes, sliders, scroll bars, control lists, control groups*, todos completamente personalizables y pertenecientes a la librería *guilib* incluida dentro de la aplicación. Dicha librería, aprovecha al máximo las prestaciones de hardware existentes actualmente para gráficos 3D, para proveer suaves animaciones y transiciones. (Nelson, 2009)

Las *skins* son definidas por ficheros XML e imágenes. Existe un fichero XML para describir cada una de las ventanas y diálogos dentro del núcleo de la aplicación, en el cual se especifican las dimensiones, visibilidad, imágenes y detalles de navegación de cada control, así como las animaciones y transiciones de la ventana. Además de los controles predefinidos, pueden agregarse y configurarse otros en cada ventana, para ejecutar funciones internas del XBMC y *scripts* de Python, levantar ventanas específicas, o requerir información del sistema. La visibilidad condicional de los controles puede basarse en la acción que ejecuta la aplicación en un momento determinado y puede ser retrasada si es necesario.

El procesador de *skins* contiene además muchas otras partes de la interfaz. Como puede apreciarse en la (Figura 2.5), existe un administrador de ventanas (*window manager*), cuya función es estar al tanto de los estados de vista de cada una de ellas y el control seleccionado en cada instante. Un administrador de audio (*audio manager*), que reproduce un sonido durante el movimiento del cursor, un *click* o una acción de retroceso.

XBMC ha sido traducido a docenas de idiomas. La librería *guilib* permite localizar todas las cadenas o *strings*, gracias a esto, los desarrolladores de *skins* pueden declarar sus propios textos con traducciones a los idiomas que estimen convenientes. Dicha biblioteca además se ocupa de la entrada de usuario, tomando eventos de un control remoto, teclado, mouse o cualquier otro dispositivo a través de un cliente de eventos, y pasándolos a la aplicación. (Nelson, 2009)

Interfaces

XBMC ofrece varias interfaces para controlar, modificar el funcionamiento y añadir contenidos. Cuenta con un intérprete de Python embebido, que provee dos de estas interfaces. La primera, los *scripts*, que no son más que pequeños programas diseñados para funcionar por sí solos dentro del XBMC. Los módulos *xbmc* y *xbmcgui* definen la API de Python dentro de la aplicación, dentro de los mismos se encuentran las funciones para el control de reproductores y manipulación de ajustes. La interfaz gráfica de los *scripts* es diseñada utilizando el módulo *xbmcgui*, con la clase WindowXML, el *script* puede tener el mismo ambiente gráfico del XBMC. (Nelson, 2009)

La segunda interfaz son los *plug-ins* o simplemente “plugins”. Los *plugins* son *scripts* especiales de Python diseñados para llenar una lista de directorios con contenidos que no

pueden ser fácilmente descritos de otro modo, organizándolos en las categorías normales del XBMC (música, videos, imágenes y programas).

El acceso remoto al XBMC es permitido gracias a un servidor web interno, que provee una página desde donde es posible controlar la sesión abierta desde un navegador web. Este servidor web brinda además la API-HTTP, para que los desarrolladores agreguen controles a sus aplicaciones. (XBMCTeam, 2012)

El Servidor de Eventos dentro del XBMC es una API cuyo trabajo es permitir a los desarrolladores añadir soporte para varios tipos de dispositivos de entrada, escucha en un *socket* UDP a los Clientes de Eventos, para conectarse y enviar eventos. Los clientes corren separadamente de XBMC y pueden ser escritos en cualquier lenguaje de programación que soporte el protocolo UDP. Su función principal es enviar las señales de los botones presionados en un control remoto, teclado u otro dispositivo de entrada al XBMC, aunque también puede enviar mensajes de notificación para eventos tales como el estado de la batería.

Existen dos nuevas APIs en desarrollo: (Nelson, 2009)

- **DLL Add-ons:** esta pretende añadir prestaciones similares al intérprete de Python embebido, con la diferencia de que las aplicaciones deberán escribirse en C o C++ y compiladas en módulos.
- **PVR Clients**: esta permitirá la interacción con aplicaciones PVR como MythTV y VDR, añadiendo al XBMC la posibilidad de reproducir TV en vivo.

Administrador de contenidos

Para que el contenido sea visible en XBMC debe crearse una fuente para poner a punto el mismo. Localmente, puede añadirse cualquier contenido desde el disco duro del usuario, CD-ROM o DVD-ROM. Pero es con las fuentes remotas donde el XBMC se pone más interesante.

Los usuarios pueden servirse del contenido existente en otro equipo utilizando SMB (para archivos compartidos de Windows), HTTP, FTP, UPnP y DAAP.

Cuando una fuente es añadida, todo su contenido está disponible inmediatamente gracias al visor de archivos. El mismo presenta el contenido en un explorador, con un icono, un

nombre de archivo, y otros detalles en dependencia del tipo de ordenamiento seleccionado. Si la fuente contiene datos estáticos (es decir, que no son *streams*), entonces dichos datos pueden ser escaneados a la biblioteca. (XBMCTeam, 2012)

Núcleo reproductor

La calidad de reproducción es el éxito mayor del XBMC. Con este fin, un par de reproductores personalizados son empleados para asegurar el máximo desempeño. (Nelson, 2009)

- ***DVD Player***: es utilizado para todas las reproducciones de video y está basado en la librería de decodificación multimedia FFMpeg. Contiene las prestaciones de los codecs de audio y video más populares, varios formatos de subtítulos, así como soporte para menú de DVD.
- ***PAP Player***: es utilizado para la reproducción de archivos de música. Utiliza muchas de las prestaciones que ofrecen los codecs de código abierto.

Compartimiento de archivos

XBMC provee el compartimiento de archivos en una red local, mediante un servidor UPnP, que permite a otros clientes UPnP dentro de la red servirse de los contenidos escaneados en la biblioteca, por ejemplo Windows Media Player, Xbox360 y Playstation3.

2.4 Python

Como ya hemos tratado, XBMC incluye un intérprete de Python que permite desarrollar *add-ons* (scripts y plugins) que interactúen fácil y limpiamente con la consola del mismo, todo sin requerir profundos conocimientos de programación.

Python es un “lenguaje de programación de alto nivel, interpretado, orientado a objetos y con semánticas dinámicas”. (Hetland, 2005)

Permite implementar la funcionalidad que el programador desee en sus programas, de forma clara y legible. Sin embargo, no resulta tan rápido como otros conocidos como C y C++, aunque con el ahorro de tiempo de programación que su uso supone, la diferencia de velocidades es un precio muy bajo a pagar y generalmente imperceptible para la mayoría de las aplicaciones. Para programadores en C, existe la posibilidad de desarrollar las partes

más críticas de los programas en este lenguaje y que las mismas interactúen con otras partes programadas en Python.

Python ha adquirido gran popularidad a nivel mundial desde su creación por Guido Van Rossum. Es utilizado extensamente en tareas de administración de sistemas (es, por ejemplo, un componente vital en muchas distribuciones de Linux), además es utilizado por instituciones reconocidas como la NASA, Yahoo y Google. (Hetland, 2005)

En este epígrafe se tratan los elementos fundamentales de este lenguaje que permiten interactuar con el software XBMC.

2.4.1 Python para XBMC

Para comenzar a programar en Python (especialmente en Python para XBMC), la forma más fácil es creando un *script*. El programa tradicional “Hello World!”, escrito como un *script* de Python para XBMC se vería de la siguiente forma:

```
print "Hello World!"
```

Es decir, se vería de la misma forma que si lo escribiéramos en la consola de Python, esto gracias al ya anteriormente mencionado intérprete embebido dentro del XBMC.

Mientras los *scripts* ofrecen flexibilidad y control total sobre la interfaz gráfica del XBMC, los *plugins* permiten rápida y consistentemente presentar información al usuario a través de la estructura de menús estándar del mismo.

Los *plugins* son *scripts* escritos en Python, pero desempeñan una función específica y deben ser salvados en una carpeta apropiada dentro de la instalación del XBMC. Son muy utilizados para anclarse a sitios web de *streaming* de videos, mostrándolos en una lista de la misma forma que se muestran los contenidos almacenados en un disco duro local. La distinción entre *scripts* y *plugins* es más teórica que práctica, pues un *plugin* puede levantar un *script* y viceversa, así como utilizar las mismas funciones. (Alex y Alexpoet, 2010)

XBMC Python es una implementación de Python 2.4 (específicamente, 2.4.5 (#71, Nov 18 2008, 18:41:39) [MSC v.1310 32 bit (Intel)]). Por este motivo cuenta con todos los módulos estándar de esta versión, los cuales se pueden listar ejecutando el siguiente script:

import sys
print sys.builtin_module_names

La descripción de otros módulos estándar puede encontrarse en el archivo “python24.zlib”, dentro de la instalación del XBMC, el cual puede ser explorado utilizando WinRAR o 7zip. (XBMCTeam, 2012)

Además de las librerías estándar, XBMC Python utiliza un grupo de librerías personalizadas, tales como: *xbmc*, *xbmcgui* y *xbmcplugin*. (Tabla 2.1)

Tabla 2.1 Descripción de librerías estándar de Python para XBMC.

Módulo	**Descripción**
xbmc	Ofrece clases y funciones que proveen información acerca de la reproducción en curso y permiten la manipulación del reproductor. Además facilita la obtención de información del sistema.
xbmcgui	Ofrece clases y funciones que manipulan la Interfaz Gráfica de Usuario a través de ventanas, diálogos y controles.
xbmcplugin	Ofrece clases y funciones que permiten a los desarrolladores presentar información a través de la estructura estándar de los menús del XBMC.

Controles gráficos disponibles

Dentro del módulo *xbmcgui* existe una amplia variedad de controles gráficos, que están disponibles para los desarrolladores de *skins*. Dichos controles son una herramienta básica para modificar y personalizar la interfaz gráfica del XBMC. A continuación se describen las características de algunos de ellos: (XBMCTeam, 2012)

- **Group Control**: Es uno de los controles más importantes. Permite agrupar otros controles, aplicando atributos a todos como si fueran uno solo. También recuerda el último botón navegado, facilitando la acción de retroceso a una posición anterior.

- **Group List Control:** es un caso especial del *group control*. Es utilizado para poner un grupo de controles en una lista (ya sea horizontal o verticalmente), permitiendo la navegación (scroll) por la misma.
- **Label Control:** es utilizado para mostrar textos en XBMC, pudiendo seleccionar la fuente, tamaño, posición y contenidos de los mismos.
- **Fade Label Control:** es usado para mostrar múltiples piezas de texto en el mismo espacio en XBMC, pudiendo seleccionar la fuente, tamaño, posición y contenidos de los mismos.
- **Image Control:** es utilizado para mostrar imágenes en XBMC. Permite seleccionar la posición, tamaño, transparencia y contenidos de la imagen a mostrar.
- **MultiImage Control**: es utilizado para reproducir una presentación de imágenes de una carpeta en XBMC. Permite seleccionar la posición y el tamaño de dicha presentación, así como el tiempo de la misma.
- **Button control:** es utilizado para crear botones *push* en XBMC. Permite elegir la posición, tamaño y aspecto de los mismos, así como la acción que ejecutarán al ser presionados.
- **Radio button control:** es utilizado para encender/apagar configuraciones.
- **Select button control:** es utilizado para crear un botón *push* que oculte múltiples opciones dentro del XBMC.
- **Toggle button control:** es utilizado para crear botones que tienen dos estados. Permite elegir la posición, tamaño y aspecto de los mismos.
- **Multiselect control:** es utilizado para crear una simple línea de texto con múltiples piezas seleccionables que pueden realizar varias acciones. Permite elegir la posición, tamaño, y aspecto del texto así como la acción a ejecutar cuando un elemento es seleccionado.
- **Spin Control:** es utilizado para cuando una lista de opciones puede ser seleccionada con los botones arriba/abajo. Permite elegir la posición, tamaño, y aspecto del spin control.

- **Scroll Bar Control:** es utilizado para el desplazamiento en todo tipo de listas. Permite seleccionar la posición, tamaño y aspecto de la barra de *scroll*.
- **Settings Slider Control:** es utilizado en pantallas de configuración donde una opción es mejor especificada cuando se utiliza una escala deslizable. Es prácticamente un cruce entre el *button control* y el *slider control*.
- **Progress Control:** es usado para mostrar el progreso de una acción que puede tomar un largo tiempo, o para mostrar la posición actual durante la reproducción de un archivo de música o video. Permite seleccionar la posición, tamaño y aspecto del *progress control*.
- **Visualization Control:** es utilizado para mostrar una visualización cuando se reproduce música con el XBMC.
- **Video Control:** utilizado para mostrar el video que se reproduce actualmente en cualquier otra parte de la Interfaz Gráfica de Usuario
- **Mover Control:** es usado para la calibración de la pantalla en XBMC. Permite elegir el tamaño y aspecto del mover control.
- **Resize Control:** usado para configurar la razón de píxeles en la calibración de la pantalla.

2.4.2 Funciones propias del sistema

Las tres librerías en la Tabla 2.1 permiten el control total sobre la Interfaz Gráfica de Usuario, pero no permiten interactuar con otras partes del sistema. Para interactuar con el sistema se utilizan funciones propias del mismo, a las que se puede acceder utilizando la función *executebuiltin* perteneciente al módulo *xbmc* de Python. A continuación describimos las utilizadas en el desarrollo de nuestro trabajo:

- **Mastermode:** Ejecuta XBMC en modo Maestro.
- **ActivateWindow (window[,dir,return]):** Abre la ventana indicada. El parámetro *window* puede ser la id de la ventana o su nombre, en caso de tratarse de una ventana estándar. Además de esto en caso de ser la ventana de Música, Video, Imágenes o Programas, el parámetro opcional *dir* definirá qué carpeta será abierta por defecto. El parámetro return indicará a XBMC que utilice la carpeta indicada en *dir*, como directorio raíz.

- **ReplaceWindow (window,dir):** Remplaza la ventana actual con la ventana dada, pero al contrario de *ActivateWindow*, no regresa a la anterior cuando la misma es cerrada.
- **RunScript (script [,args]*):** Corre *scripts* de Python. Debe especificarse el camino completo hacia el script. Todos los parámetros extras son pasados al *script* como argumentos y pueden ser accedidos por Python usando sys.argv
- **RunAddon ():** Corre el plugin/script especificado.
- **RunPlugin (plugin):** Corre el plugin especificado, requiere introducir el camino completo hacia el mismo.
- **RecursiveSlideShow (dir):** Corre una presentación de imágenes del directorio especificado incluyendo todos los subdirectorios.
- **PlayerControl (command):** Permite el control de música y videos. El commando (command) puede ser Play, Stop, Forward, Rewind, Next, Previous, BigSkipForward, BigSkipBackward, SmallSkipForward, SmallSkipBackward, Random, RandomOn, RandomOff, Repeat, RepeatOne, RepeatAll, RepeatOff, Partymode(music), Partymode(video) o Partymode(path to .xsp file), and Record.
- **Playlist.PlayOffset:** Comienza la reproducción desde una posición particular en la lista de reproducción.
- **Skin.SetNumeric (numeric[,value]):** Levanta un diálogo de teclado y permite al usuario entrar un dato numérico.
- **Skin.SetPath (string[,value]):** Levanta un buscador de carpetas y permite al usuario seleccionar una carpeta de imágenes para ser usadas con el control *multimage* del *skin*.
- **Skin.SetImage (string[,value]):** Levanta un buscador de archivos que permite al usuario seleccionar un archivo de imagen para ser utilizado con el control *image* del *skin*.
- **Skin.SetAddon (string,type):** Levanta el diálogo indicado y permite al usuario seleccionar un add-on del tipo dado para ser utilizado en el *skin*.

2.4.3 Script de Python para XBMC

A continuación se muestra un pequeño ejemplo de un *script* escrito en Python para XBMC, las líneas con el símbolo # serán tomadas por el intérprete como comentarios. (Alex y Alexpoet, 2010)

La función del *script* es crear una ventana con una imagen de fondo, utilizando el ambiente gráfico del XBMC.

```
# importamos las librerías gráficas de XBMC, así podremos utilizar sus controles y
funciones
import xbmc, xbmcgui
# nombramos y creamos nuestra ventana
class BlahMainWindow(xbmcgui.Window):
   # y la definimos como self
   def __init__(self):
   # añadimos un control de imagen a nuestra ventana
   self.addControl(xbmcgui.ControlImage(0,0,720,480,
'Q:\\scripts\\background.jpg'))
# almacenamos la ventana como una variable short para su uso más fácil
W = BlahMainWindow()
# Corremos la ventana con background que hemos creado
W.doModal()
# después que la ventana es cerrada, la destruimos
del W
```

2.5 XML

XML son las siglas de Extensible Markup Language, es un lenguaje utilizado para describir documentos y datos en un formato estandarizado basado en texto, que puede ser fácilmente transportado por los protocolos estándar de Internet. En este epígrafe se describen los aspectos básicos de este lenguaje, para asegurar la comprensión de los archivos de configuración del XBMC.

2.5.1 Elementos y atributos

Como XML es diseñado para describir datos y documentos, la Recomendación para XML de W3C, es muy estricta acerca de un pequeño núcleo de requerimientos de formato que hacen la diferencia entre un documento de texto y un documento XML actual. Siguiendo tal recomendación, los documentos XML pueden contener elementos, atributos y texto. (Harold, 2001)

Los elementos se describen de la forma siguiente: (Benz y Durant, 2003)

<element></element>

Como puede apreciarse, existe un rótulo de inicio y uno de fin, ambos encierran al elemento a ser formateado. Existen algunas reglas básicas para elementos de documentos XML. Sus nombres pueden contener letras, números, guiones, subguiones pero no pueden contener espacios, por este motivo se utilizan los subguiones para sustituirlos. El nombre de

un elemento puede comenzar con una letra, un subguión o dos puntos, pero nunca con caracteres no alfabéticos, números o las letras xml.

Los atributos contienen valores asociados a un elemento.

<element attribute=”value”></element>

Las reglas básicas para los elementos aplican también para los atributos, con algunas adicionales. El nombre de un atributo debe seguir al nombre de elemento, luego un signo de igualdad (=) y por último el valor del atributo encerrado en comillas simples o dobles.

El texto debe situarse entre el rótulo de abrir y el de cerrar de un elemento, y normalmente representa el dato actual asociado con el mismo.

<element attribute=”value”>text</element>

El texto no está regido por las mismas reglas de sintaxis de los elementos y atributos, por eso cualquier texto puede ser almacenado entre elementos de un documento XML. (Benz y Durant, 2003)

2.5.2 Estructura de un archivo XML

Los elementos, atributos y texto en los documentos XML, deben regirse por una estructura organizativa, para analizarla utilicemos un pequeño ejemplo de un documento XML 1.0.

```
<?xml version=”1.0” encoding=”UTF-8”?>
<rootelement>
<firstelement position=”1”>
<level1 children=”0”>nivel 1 de elementos anidados</level1>
</firstelement>
<secondelement position=”2”>
<level1 children=”1”>
<level2>nivel 2 de elementos anidados</level2>
</level1>
</secondelement>
</rootelement>
```

La mayoría de los documentos XML comienzan con un elemento <?xml?> al inicio de la página, lo que se conoce como declaración del documento XML. Esta declaración es opcional, pero muy útil para determinar la versión de XML y el tipo de codificación de la fuente de datos, la más común es:

<?xml version=”1.0” encoding=”UTF-8”?>

2.5.3 Archivos XML dentro del XBMC

Ya hemos tratado en epígrafes anteriores que cada ventana del ambiente gráfico en XBMC está descrita por un archivo XML, mediante el cual es posible modificar y agregar nuevos controles. Los archivos XML dentro de la instalación del XBMC permiten interactuar con los controles propios de las librerías gráficas, como *labels, buttons, radio buttons, text boxes, spin boxes, sliders, scroll bars, control lists, control groups,* configurándolos para obtener el nivel de personalización que deseemos. Analizamos a continuación un ejemplo del uso y configuración del control *group*, utilizando el lenguaje XML.

```
<control type="group" id="17">
    <description>My first group control</description>
    <posx>80</posx>
    <posy>60</posy>
    <width>250</width>
    <height>30</height>
    <defaultcontrol>2</defaultcontrol>
    <visible>true</visible>
    <onup>2</onup>
    <ondown>3</ondown>
    <onleft>1</onleft>
    <onright>1</onright>
    ... more controls go here ...
</control>
```

Nótese como cada *tag* se corresponde con una propiedad configurable del control, por ejemplo el *tag defaultcontrol*, especifica el control que será enfocado por defecto cuando el grupo es enfocado.

Todos los controles dentro de un grupo, toman sus posiciones relativas a la posición del grupo. Por tanto, cada grupo requiere que sean definidos los atributos <posx>, <posy>, <width>, y <height>, de faltar alguno de los mismos, su valor será heredado del control group superior en caso de existir o de la ventana principal. Para mayor comprensión de este aspecto describimos brevemente un ejemplo de un primer grupo dentro de una ventana de pantalla completa, utilizando la norma PAL (720x576). (XBMCTeam, 2012)

```
<control type="group" id="15">
  <posx>30</posx>
  <posy>70</posy>
  <width>400</width>
  ... more controls go here ...
```

</control>

Como podemos apreciar al no definirse el atributo <height>, XBMC automáticamente lo hace igual a 506, ya que hereda el <height> estándar de la ventana de 576 y le resta el valor especificado por el atributo <posy>.

2.6 Personalización de la interfaz gráfica del software XBMC

En este epígrafe se describen los pasos seguidos para la implementación de la interfaz gráfica (Figura 2.6), personalizada para que el usuario interactúe con los contenidos de una forma amigable.

Figura 2.6 Estructura de la interfaz gráfica deseada.

Para utilizar las características acogedoras del ambiente gráfico del software XBMC, se han mantenido las texturas y estructuras de la mayoría de las ventanas de su *skin* por defecto llamado “Confluence”, modificando solo las necesarias para adaptar la interfaz de uso

general del software, a una que cumpla con los requerimientos de los servicios de IPTV. La ventana *Home* es la que ha sufrido mayores modificaciones, pues controla el acceso a todas las prestaciones de la aplicación. La modificación de la ventana *Home* y por tanto, la implementación de la estructura deseada, se efectuó actuando sobre el archivo home.xml perteneciente al *skin*. El código XML que conforma las áreas que se describen a continuación, puede ser consultado en el **Anexo II** al final de este informe.

En el **Anexo III** puede encontrarse una imagen de la interfaz terminada.

2.6.1 Área de botones de aplicación

En esta área de la interfaz se encuentran los botones de acceso a las ventanas de contenidos del software XBMC (Pictures, Music, Videos), y constituye el elemento fundamental de la pantalla principal de la aplicación. Los botones se han distribuido dentro de una lista de menú, con los siguientes nombres y funciones.

Pictures: Permite acceder a la ventana de imágenes. En esta ventana se encontrarán los archivos de imagen previamente agregados desde una fuente.

Music: Permite acceder a la ventana de música, dentro de la cual aparecerán listados todos los archivos de música agregados previamente desde una fuente.

Videos: Permite acceder a la ventana de videos. En esta ventana se encontrarán los archivos de video que hayan sido agregados previamente desde una fuente.

Internet: Este botón tiene características especiales, debido a que su función no es acceder a ninguna de las ventanas de contenidos de la aplicación; al ser presionado ejecuta una función interna del sistema que interpreta un *script* de Python. Dicho *script* ha sido programado para ejecutar un navegador web. La función utilizada para que el este botón funcione de la forma descrita puede revisarse en el código XML mostrado en el **Anexo I**.

El script de Python utilizado puede consultarse en el **Anexo II** al final de este informe.

2.6.2 Área de botones de configuración y otras opciones

Dentro de esta área se encontrarán los botones de acceso a las opciones de configuración de la aplicación y de apagado del sistema. Los mismos han sido implementados con los siguientes nombres, propiedades y funciones.

Settings: Este botón se ha creado sin etiqueta de texto. Permite el acceso a la ventana de configuración de la aplicación. Tiene propiedades específicas, debido a que no es de nuestro interés que el usuario tenga acceso al mismo, por tanto su visibilidad se ha condicionado. El botón *Settings* solo estará visible cuando el sistema esté en modo master, al que solo podrá acceder el personal de servicios técnicos, introduciendo una contraseña. Esta configuración puede consultarse en el código mostrado en el **Anexo I**.

Botón Favoritos: Permitirá acceder a la ventana Favoritos del XBMC, donde aparecerán listados los contenidos previamente seleccionados por el usuario como favoritos. Este botón no se ha modificado.

Botón de apagado: Permite acceder a un grupo de opciones, como Apagar, Reiniciar y Entrar al Modo Master. Este botón muestra una serie de opciones diferentes cuando se entra el sistema en modo *master*, por ejemplo la opción *exit* que permite salir del ambiente gráfico de la aplicación XBMC y entrar a la consola del sistema operativo subyacente Ubuntu 10.04.

2.6.3 Área de backgrounds

El área de backgrounds se ha reservado para personalizar las ventanas de nuestra interfaz con imágenes de fondo. Las imágenes que hemos utilizado para tal fin, se han situado en la carpeta de nombre backgrounds dentro de la instalación del *skin*, desde donde son llamadas utilizando el código XML.

2.7 Conformación del set-top box personalizado

Para conformar el producto final se instaló el software XBMC modificado en una computadora, con las siguientes características:

- Procesador Celeron ® 430 1.8 GHz
- 1 GB de Memoria RAM,
- 250 GB de disco duro
- Placa madre DG41TI Intel
- Tarjeta de red Realtek 10/100/1000

Se ha utilizado esta máquina de regulares prestaciones, como equivalente a otros tipos de hardware existentes en el mercado con los que se puede lograr un producto más terminado.

Para la instalación del software se utilizó la versión 10.1 del XBMC distribuida en Live CD, la cual cuenta con el sistema operativo subyacente Ubuntu 10.04. Luego se realizaron las siguientes acciones:

- Instalación de un navegador web para el servicio de Internet, desde el repositorio de Ubuntu 10.04.
- Instalar y habilitar el skin que contiene todas las modificaciones realizadas al software.

2.7.1 Conclusiones del capítulo

En este capítulo se han analizado las características fundamentales del software XBMC debido a que fue el escogido para el desarrollo del trabajo. Se han tratado además los lenguajes de programación utilizados para interactuar con dicho software y por último se han implementado los cambios requeridos en el mismo para el cumplimiento de los objetivos propuestos.

CAPÍTULO 3. RESULTADOS Y DISCUSIÓN

En este capítulo se describen las pruebas experimentales realizadas para validar el correcto funcionamiento del producto terminado, en un escenario real. Se ha utilizado el software de monitoreo Munin para medir el consumo de recursos de hardware y la estabilidad del mismo durante la explotación de todas sus prestaciones.

3.1 Conformación del escenario de prueba

El escenario de prueba (Figura 3.1), estuvo formado por tres componentes principales.

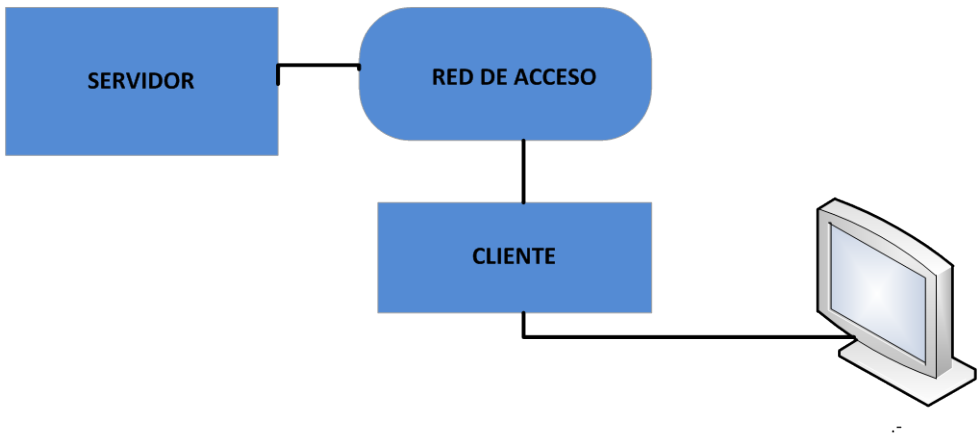

Figura 3.1 Estructura del escenario de prueba.

Servidor

Como servidor, se utilizó una máquina de propósito general, en la cual se instaló la aplicación de código abierto VLC (Video Lan Client).

La aplicación VLC es una solución de software para el *streaming* de video, desarrollada por estudiantes de la Ecole Centrale Paris y desarrolladores de otras partes del mundo, bajo la licencia GNU General Public License. Puede difundir *streams* de archivos MPEG-1,

MPEG-2 y MPEG-4, DVDs, canales digitales satelitales, canales digitales terrestres y videos en vivo sobre la red, utilizando Unicast o Multicast, o ser utilizada como cliente para recibir, decodificar y reproducir streams. (Lattre et al., 2005)

El servidor se instaló en la cabecera de la red HFC, en el Cayo Santa María y se conectó a la red de cable.

Cliente.

Como cliente se utilizó el equipo terminado, ya descrito en el capítulo anterior.

Red de acceso.

La interconexión del cliente con el servidor se efectuó como muestra la Figura 3.2, utilizando la red HFC existente el polo turístico de la Cayería Norte de Villa Clara. El servidor, se situó en la cabecera y se conectó al CMTS, mientras que el cliente, se conectó a uno de los CMs instalados.

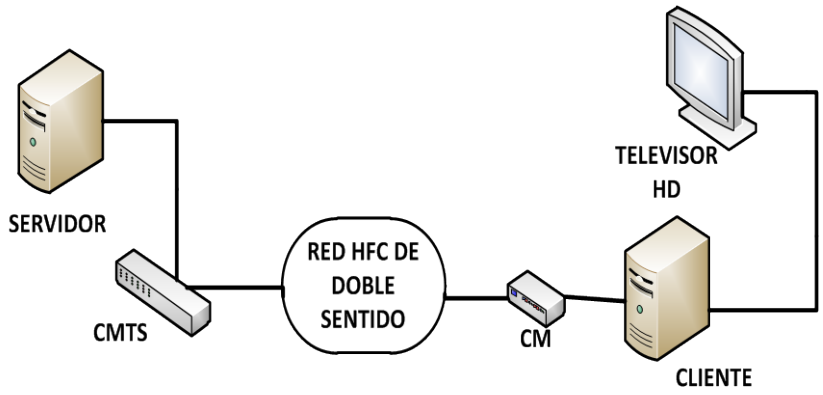

Figura 3.2 Esquema de conexión del cliente y el servidor a través de la red HFC.

Los equipos utilizados cumplen específicamente con la especificación DOCSIS 3.0.

3.2 Comprobación del sistema en el escenario creado.

Para la comprobación del set-top box, se monitoreó el funcionamiento del mismo durante varios días en el escenario creado, tiempo en el cual, se mantuvo el acceso a servicios de *streaming* y se habilitó la navegación web, para poder medir el consumo de recursos de hardware se explotan todas las prestaciones del equipo.

En este epígrafe se describen los aspectos de configuración de las herramientas de software utilizadas en la comprobación. Además, se realiza un análisis crítico de los resultados obtenidos.

3.2.1 Configuración del VLC para las pruebas de streaming

Para mantener el suministro de contenidos durante todo el período de prueba, se utilizó una herramienta del software VLC que permite el *streaming* desde diferentes fuentes a la vez, llamada Video LAN Manager (VLM). Esta herramienta, permite crear nuestra propia programación para broadcast y soporta también el video bajo demanda. La programación se conformó de tal forma que estuvieran presentes todas las opciones de difusión que pueden ser reconocidas por nuestro cliente, para lo cual se utilizó el configurador VLM (Figura 3.3).

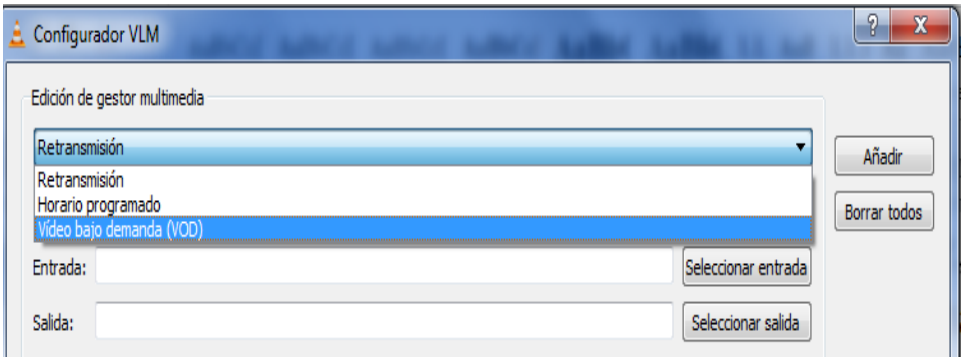

Figura 3.3 Ventana de configuración de la herramienta VLM.

Como puede apreciarse, el configurador permite seleccionar una de tres opciones para difundir el contenido. Se seleccionaron las entradas para cada tipo de *stream*, conformando la lista del administrador de multimedia (Figura 3.4). La salida se configuró de la siguiente forma:

- Contenidos que se difunden bajo demanda.
- Contenidos difundidos por broadcast.

En cada uno de los casos se utilizó el protocolo RTSP para el *streaming* variando el tipo de codificación de audio y video.

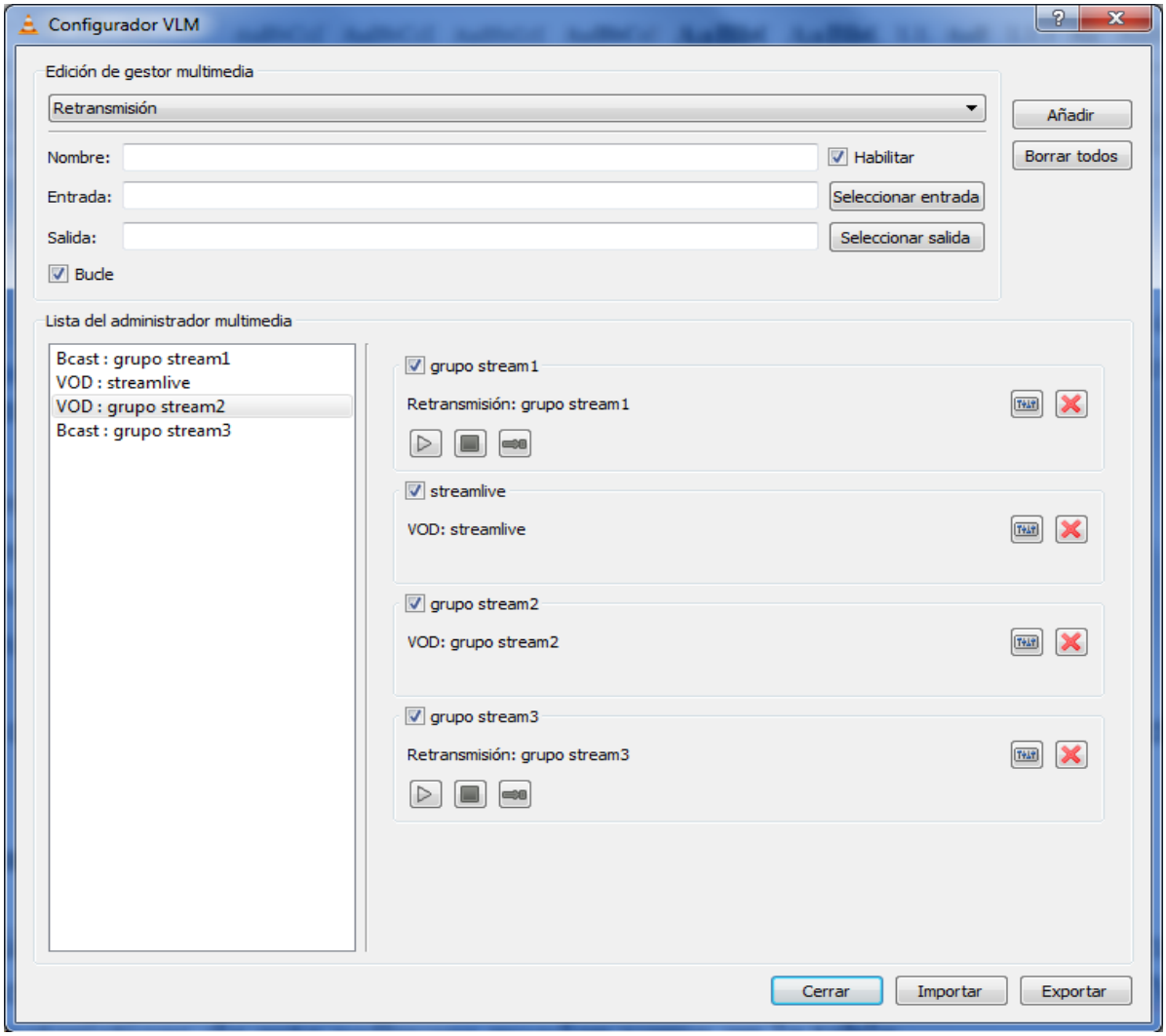

Figura 3.4 VLM configurado para el streaming de varios contenidos.

3.2.2 Configuración del cliente

Para configurar el cliente, se crearon archivos de extensión .strm, los cuales son utilizados por el XBMC para conocer la dirección de las fuentes de *stream*. Estos archivos fueron ubicados en un servidor FTP y añadidos como fuente de contenidos a las listas de reproducción del XBMC, siguiendo los pasos mostrados a continuación:

- Click en el botón Videos del menú principal y luego se accedió al menú “Add Video Source”(Figura 3.5).

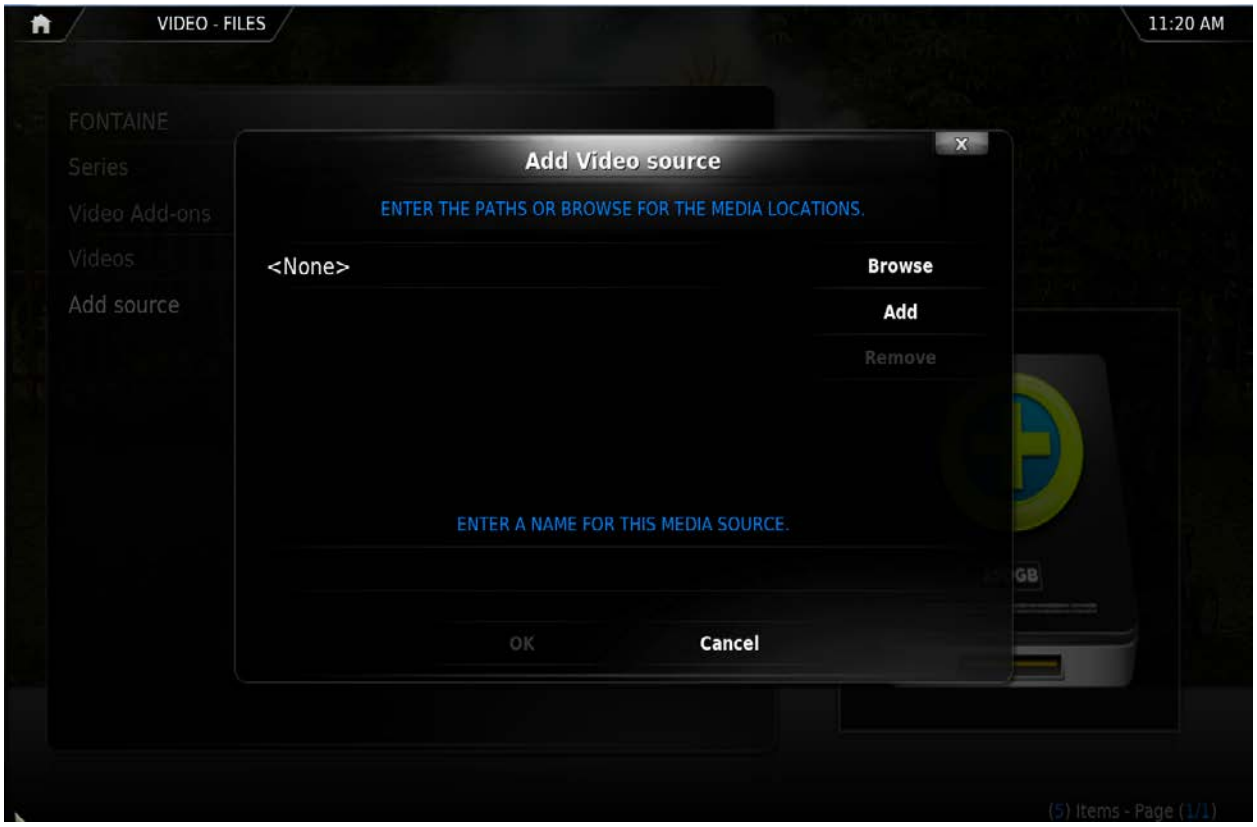

Figura 3.5 Menú para adición de fuente de contenidos.

- Utilizando la opción “Browse”, se añadió la dirección de red del servidor FTP (Figura 3.6) para localizar los archivos .strm, correspondientes a cada uno de los grupos de *streams* creados en la programación del VLC.

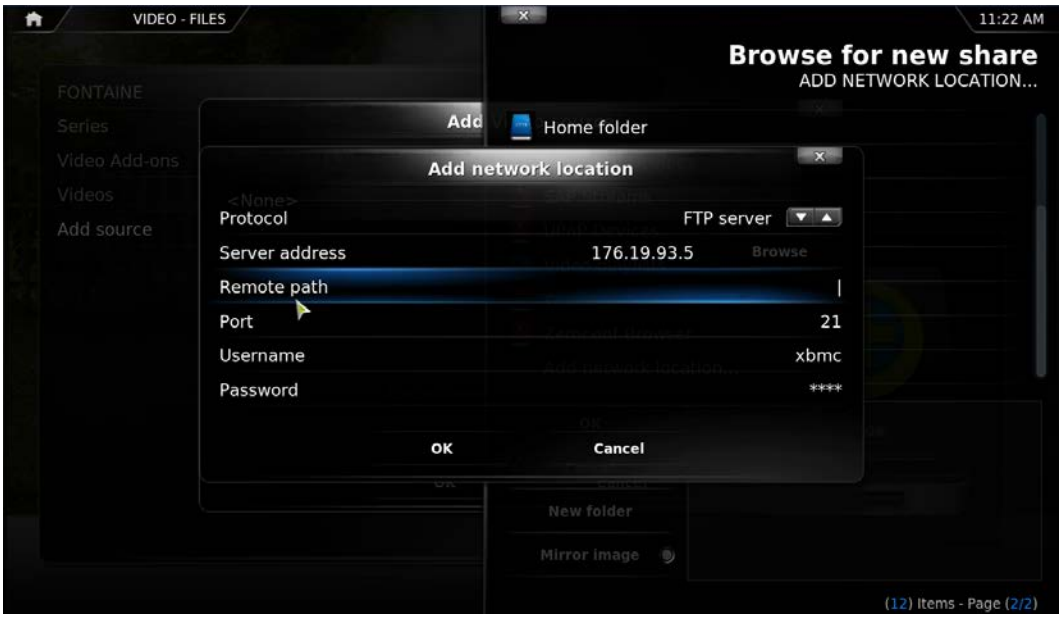

Figura 3.6 Introducción de los datos del servidor FTP.

Finalmente aparecen listados los *streams* disponibles, en la ventana videos del XBMC. Como puede apreciarse en la Figura 3.7, el nombre de los archivos aparece tal y como se define, por tanto, es necesario tener en cuenta que para lograr una mayor satisfacción del cliente, los nombres de los archivos .strm deben coincidir con los nombres de los contenidos a los que hacen referencia.

Figura 3.7 Lista de streams en el menú de reproducción del XBMC.

3.2.3 Software de monitoreo

Para realizar un análisis del consumo de recursos cuando el set-top box está en funcionamiento, se seleccionó el software Munin. Algunas de las características de éste pueden verse en la Tabla 3.1.

Tabla 3.1 Principales ventajas y desventajas de la aplicación Munin. (Borbor y Loaiza, 2009)

Ventajas	Desventajas
- Fácil de configurar e integrar, tanto en un solo host como en una red, plana o compleja. - Orientado a la autoconfiguración (munin-node-configure). - Diseñado para ser simple de extender a través de plugins. - Presenta información histórica que permite analizar tendencias, pero no permite a un usuario remoto ejecutar código (genera HTML estático) - Hace monitoreo periódico. (cada cinco minutos)	- Recibe la información sin autentificación y en texto plano, lo cual no es adecuado si existe información confidencial o sensible - Sirve para recolectar datos estadísticos, pero no funciona como herramienta para emitir alertas.

Munin se divide en tres componentes principales: (Borbor y Loaiza, 2009)

- **Servidor:** Corre en todas las máquinas monitoreadas, por defecto en el puerto 4949. Su función es configurar y llamar a los Plugins.
- **Plugins**: Cada uno de los agentes de recolección de datos que son invocados por Munin. Son también capaces de describir su función y configuración. (Figura 3.8)
- **Cliente:** Proceso que corre periódicamente (normalmente cada 5 minutos) desde un nodo central, interrogando a cada uno de los servidores y generando las páginas Webs con los resultados.

```
0 gwolf@malenkaya[1]~$ /usr/sbin/munin-node-configure
Plugin                  | Used | Extra information
------                  | ---- | -----------------
acpi                    | yes  |
apache_accesses         | no   |
apache_processes        | no   |
apache_volume           | no   |
apt                     | no   |
apt_all                 | no   |
courier_mta_mailqueue   | no   |
courier_mta_mailstats   | no   |
courier_mta_mailvolume  | no   |
                        (...)
hddtemp_smartctl        | no   |
if_                     | yes  | eth1 eth2
if_err_                 | no   |
```

Figura 3.8 Configuración de los plugins de Munin.

3.2.4 Análisis de resultados

Las capacidades de Munin para recolectar datos de los diferentes componentes de una PC son prácticamente inagotables. En nuestro caso solo utilizaremos aspectos de interés que demuestren el correcto funcionamiento de la aplicación seleccionada como componente de software sobre el equipo utilizado como cliente. De esta forma se comprobará que la misma podrá ser portada a otro tipo de hardware con características similares, pero que ofrezca un mejor acabado al producto final. Se analizarán entonces, parámetros tales como: carga, uso de CPU y uso de memoria.

En el caso de la carga promedio del cliente (Figura 3.9) se puede observar que el comportamiento es estable en el caso del día seleccionado, primeramente por debajo de un valor de 1 mientras el sistema está inactivo, luego por debajo de 1.6 mientras el cliente accede al servicio de *video streaming*, y por último puede apreciarse un incremento en la carga promedio hasta alcanzar un valor de 2.2 durante las últimas seis horas aproximadamente, cuando fue utilizado simultáneamente con la reproducción de contenidos, el servicio de navegación web. Por otra parte, la gráfica semanal muestra que la mayor carga ocurre precisamente el día seleccionado, lo cual está relacionado con el acceso a más de un servicio simultáneamente (reproducción de contenidos y navegación web).

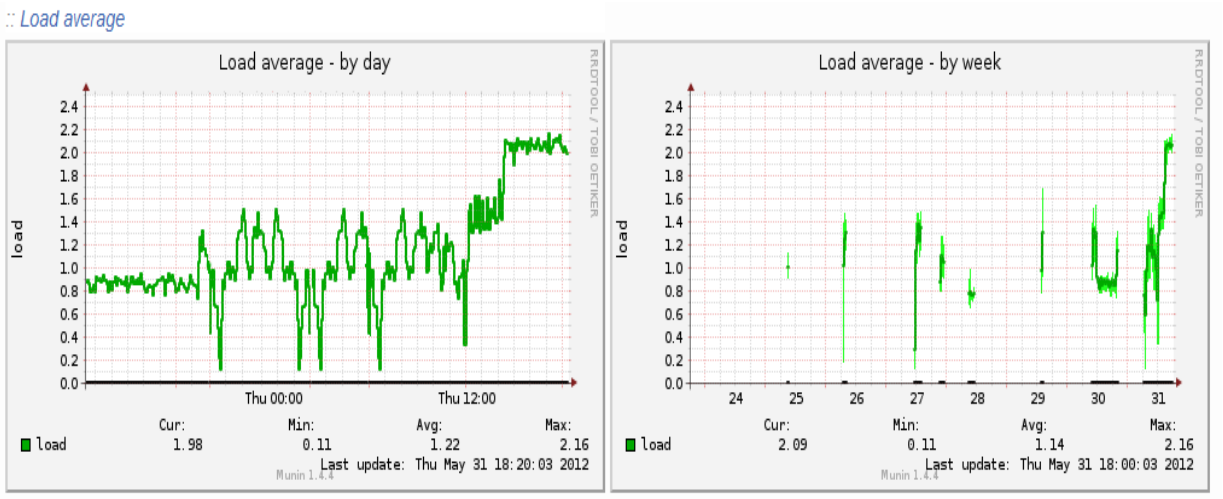

Figura 3.9 Carga promedio.

Al analizar el uso de la CPU es importante destacar que los resultados obtenidos para el hardware utilizado, serán completamente aplicables para otros tipos de hardware de iguales prestaciones o superiores. Teniendo en cuenta esto puede apreciarse en la Figura 3.10 que la CPU está funcionando en el peor de los casos al 30% de su capacidad, lo cual indica que puede soportar mucha más carga de trabajo sin ver afectado su rendimiento máximo.

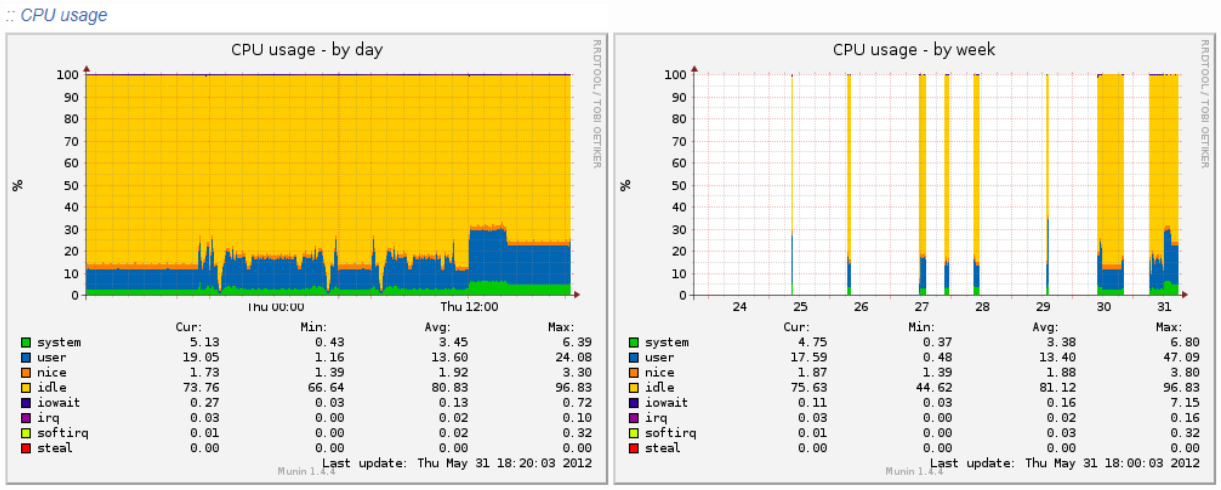

Figura 3.10 Uso de la CPU.

Por último tenemos la Figura 3.11, la cual nos muestra el uso de memoria RAM, como podemos ver existen picos cercanos a los 600 MB en determinados momentos, pero estos valores están muy por debajo de las capacidades de las memorias RAM existentes en el

mercado, por tanto no es un factor por el que preocuparse a la hora de instalar nuestro sistema en otros tipos de hardware similares o superiores al utilizado.

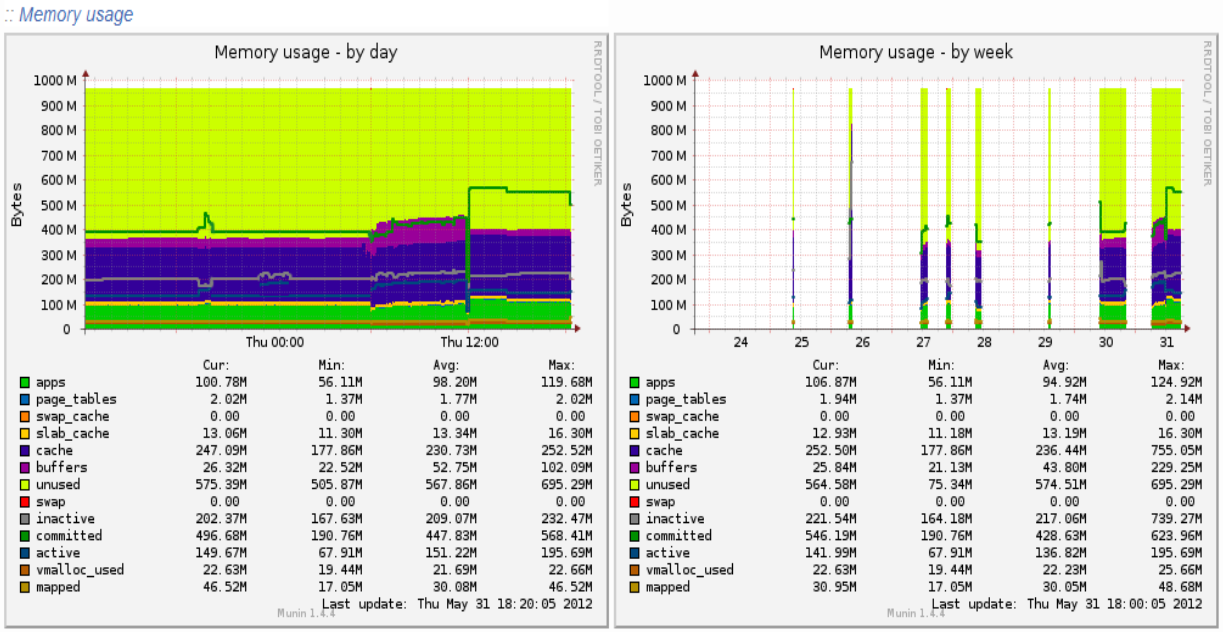

Figura 3.11 Uso de memoria.

En cada una de las gráficas semanales pueden apreciarse zonas no cubiertas, lo cual se debe a los períodos de tiempo durante la comprobación, en los que el *set-top box* estuvo apagado.

3.2.5 Conclusiones del capítulo

En este capítulo se han descrito las pruebas realizadas al equipo creado, en un escenario real. Se han aclarado los pasos seguidos para la instalación y configuración del mismo. Por último se han recogido y analizado las gráficas del comportamiento histórico durante su explotación, mediante las cuales se ha validado su correcto funcionamiento y la posibilidad de ser portado a otros tipos de hardware de similares características.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones

1. Aunque existen diferentes tipos de sistemas de IPTV en el mercado internacional, los precios de los mismos son demasiado altos para implementarlos en nuestro país, por lo que encontrar alternativas viables, continua siendo una necesidad.
2. La aplicación XBMC dentro del amplio mundo del software libre, es una de las plataformas de gestión de contenido multimedia más robustas e integrales que existen en estos momentos.
3. La posibilidad de interactuar con el software XBMC utilizando el lenguaje de programación Python constituye una ventaja, pues es actualmente uno de los más utilizados en el mundo, por lo que la actualización del sistema se hará más fácil.
4. La implementación en el software XBMC de un servicio de navegación web, permitirá el acceso a Internet utilizando una interfaz común con el resto de los servicios.
5. Se comprobó el correcto funcionamiento en un escenario real, del equipo obtenido como resultado de la instalación del software XBMC modificado, en un hardware de prestaciones regulares.
6. Se observó la gran potencialidad que poseen los programas de monitoreo y análisis histórico de datos estadísticos, para comprobar el comportamiento de los sistemas mientras están en uso, facilitando de esa forma realizar comparaciones objetivas de su desempeño con otros productos similares.

Recomendaciones

Una vez concluido el trabajo se proponen las siguientes recomendaciones:

1. Profundizar el estudio en el tema, buscando la perfección del sistema propuesto hasta el punto en que se convierta en una herramienta reconocida, que se integre con aquellas soluciones que se aplican de forma aislada en nuestro país.
2. Continuar con la investigación para implementar mecanismos de autenticación de los usuarios dentro del sistema, así como de tarificación de los servicios brindados a través del mismo.
3. Continuar mejorando la interfaz gráfica de usuario del sistema, de forma tal que los contenidos se muestren de una forma más atractiva para el usuario.
4. Utilizar un hardware que permita obtener un mejor acabado del producto final.

REFERENCIAS BIBLIOGRÁFICAS

ALEX & ALEXPOET. 2010. *XBOX Python Tutorial*.

APOSTOLOPOULOS, J. G., TAN, W.-T. & WEE, S. J. 2002. Video Streaming: Concepts, Algorithms, and Systems.

BENZ, B. & DURANT, J. R. 2003. XML Programming Bible. New York: Wiley Publishing Inc

BORBOR, W. & LOAIZA, A. 2009. Instalación y configuración de software Open Source para monitorear el servicio y la carga de un sistema Asterik.

BRUIN, R. D. & SMITS, J. 1999. Digital Video Broadcasting: Technology, Standards, and Regulations. *In:* HOUSE, A. (ed.). Eindhoven

CABLE TELEVISION LABORATORIES, I. 2010. Physical Layer Specification. Data Over Cable Service Interface Specifications DOCSIS 3.0 ed. USA.

CABLELABS. 2006. *Issues DOCSIS 3.0 Specification Enabling 160 Mbps* [Online]. Available: http://www.cablelabs.com [Accessed 20/2/2012].

DOWNEY, J. J. 2006. DOCSIS 3.0 Overview. *Cisco Systems, Inc.*

ERDOGAN, E. 2004. *An On-Demand Advertising Model For Interactive Television.* Masters Project, Georgia Institute of Technology.

FENG, P. 2001. Digital Television Terrestrial Broadcasting Primer. *TechOnLine.*

GARCÍA, I. P. 2004. Counicación y publicidad en Televisión Interactiva. *Investigación y Marketing.* Madrid.

HAROLD, E. R. 2001. XML Bible. New York: Hungry Minds, Inc.

HELD, G. 2007. Understanding IPTV. New York: Auerbach Publications.

HETLAND, M. L. 2005. Beginning Python from Novice to Professional. *In:* GILMORE, J. (ed.).

HJELM, J. 2008. Why IPTV? *Interactivity, Technologies and Services.* John Wiley and Sons, Ltd.

KOZAMERNIK, F. & VERMAELE, L. 2005. Will broadband TV shape the future of broadcasting? .

LATTRE, A. D., BILIEN, J., DAOUD, A., STENAC, C., CELLERIER, A. & SAMAN, J.-P. 2005. VideoLAN Streaming Howto.

LÓPEZ, J. L. A. & BADIA, E. F. 2008. Streaming de video y audio.

NELSON, T. 2009. XBMC Architecture Summary.

O'DRISCOLL, G. 2008. Next generation IPTV services and technologies. Hoboken, New Jersey: John Wiley & Sons, Inc.

ORTIZ, J. P. Q. & SERNA, C. A. C. 2006. *Evaluación de servidores de streaming de video orientados a dispositivos móviles.*, UNIVERSIDAD DE ANTIOQUIA.

PULARIKKAL, B. 2009. Introduction to DOCSIS 3.0.

READ, K. & SCHAFER, S. 2009. Explorando alternativas a MythTV Cine en Casa.

SCIARA, D. R. 2004. Fundamentos de Video Streaming.

TOOLEY, M. & BOWMAN, D. 2010. An overview of the DOCSIS (Cable Internet) platform.

VECIMA NETWORKS 2008. M-CMTS & DOCSIS 3.0 Standards Overview. *Vecima Networks.*

XBMCTEAM. 2012. *XBMC Wiki* [Online]. Available: http://wiki.xbmc.org [Accessed 25/1/2012].

ANEXOS

Anexo I Segmento del código modificado en el archivo home.xml

```
<!--Código XML del menú principal de la interfaz gráfica personalizada-->
<control type="group">
    <animation effect="slide" end="-400,0" time="500" tween="quadratic" easing="out">WindowClose</animation>
    <animation effect="slide" start="-400,0" time="500" tween="quadratic" easing="out">WindowOpen</animation>
    <!--Textura de fondo del menu principal-->
    <control type="group">
        <animation effect="slide" end="-228,0" time="300" tween="quadratic" easing="out">WindowClose</animation>
        <control type="image">
            <posx>379</posx>
            <posy>0</posy>
            <width>1</width>
            <height>720</height>
            <texture>HomeBladeSub_Part1.png</texture>
            <animation effect="zoom" start="379,0,1,720" end="379,0,229,720" time="300" tween="quadratic"
            easing="out" condition="ControlGroup(9001).HasFocus">Conditional</animation>
        </control>
        <control type="image">
            <posx>380</posx>
            <posy>0</posy>
            <width>12</width>
            <height>720</height>
            <texture>HomeBladeSub_Part2.png</texture>
            <animation effect="slide" end="228,0" time="300" tween="quadratic" easing="out"
            condition="ControlGroup(9001).HasFocus">Conditional</animation>
        </control>
    </control>
```

```
        <control type="image">
            <posx>380</posx>
            <posy>0</posy>
            <width>12</width>
            <height>720</height>
            <texture>HomeBladeSub_Part2.png</texture>
            <animation effect="slide" end="228,0" time="300" tween="quadratic" easing="out"
            condition="ControlGroup(9001).HasFocus">Conditional</animation>
        </control>
    </control>

    <control type="image">
        <posx>-450</posx>
        <posy>0</posy>
        <width>846</width>
        <height>720</height>
        <colordiffuse>DAFFFFFF</colordiffuse>
        <texture>HomeBlade_bottom.png</texture>
    </control>

    <control type="button" id="8999">
        <description>Run Recently added</description>
        <posx>-20</posx>
        <posy>-20</posy>
        <width>1</width>
        <height>1</height>
        <label>-</label>
        <font>-</font>
        <onfocus>XBMC.RunScript(script.recentlyadded,limit=4)</onfocus>
        <onfocus>SetFocus(9000)</onfocus>
        <texturenofocus>-</texturenofocus>
        <texturefocus>-</texturefocus>
        <visible>Skin.HasSetting(homepageHideRecentlyAdded)</visible>
    </control>

    <control type="button" id="8999">
        <description>Don't run Recently added</description>
        <posx>-20</posx>
        <posy>-20</posy>
        <width>1</width>
        <height>1</height>
        <label>-</label>
        <font>-</font>
        <onfocus>SetFocus(9000)</onfocus>
        <texturenofocus>-</texturenofocus>
        <texturefocus>-</texturefocus>
        <visible>!Skin.HasSetting(homepageHideRecentlyAdded)</visible>
    </control>
```

```
<control type="fixedlist" id="9000">
    <hitrect x="50" y="50" w="330" h="620" />
    <posx>-20</posx>
    <posy>0</posy>
    <width>400</width>
    <height>720</height>
    <onleft>10</onleft>
    <onright>9001</onright>
    <onup>9000</onup>
    <ondown>9000</ondown>
    <pagecontrol>-</pagecontrol>
    <scrolltime>300</scrolltime>
    <focusposition>3</focusposition>
    <movement>2</movement>
    <animation effect="fade" start="0" end="100" time="200">Visible</animation>
    <animation effect="fade" start="100" end="30" time="200"
    condition="Window.IsVisible(1113) | ControlGroup(9001).HasFocus | ControlGroup(10).HasFocus
    | Control.HasFocus(8000) | Control.HasFocus(8001)">conditional</animation>
```

```
        <focusedlayout height="102" width="380">
            <control type="label">
                <posx>380</posx>
                <posy>21</posy>
                <width>380</width>
                <height>40</height>
                <font>font50caps_title</font>
                <textcolor>white</textcolor>
                <align>right</align>
                <aligny>center</aligny>
                <animation effect="zoom" start="65" end="100" center="380,51" time="200">Focus</animation>
                <animation effect="zoom" start="100" end="65" center="380,51" time="200">UnFocus</animation>
                <label>$INFO[ListItem.Label]</label>
            </control>

        </focusedlayout>
```

```
        <itemlayout height="102" width="380">
            <control type="label">
                <posx>380</posx>
                <posy>21</posy>
                <width>380</width>
                <height>40</height>
                <font>font50caps_title</font>
                <textcolor>grey3</textcolor>
                <align>right</align>
                <aligny>center</aligny>
                <label>$INFO[ListItem.Label]</label>
                <animation effect="zoom" start="65" end="65" center="380,51" time="0"
                condition="true">Conditional</animation>
                <animation effect="fade" start="100" end="30" time="200"
                condition="Window.IsVisible(1113) | ControlGroup(9001).HasFocus
                | ControlGroup(10).HasFocus">conditional</animation>
            </control>
        </itemlayout>
```

```
        <content>
            <item id="4">
                <label>1</label>
                <onclick>ActivateWindow(Pictures)</onclick>
                <icon>special://skin/backgrounds/pictures.jpg</icon>
                <thumb>$INFO[Skin.String(Home_Custom_Back_Pictures_Folder)]</thumb>
                <visible>!Skin.HasSetting(HomeMenuNoPicturesButton)</visible>
            </item>
            <item id="3">
                <label>2</label>
                <onclick>ActivateWindow(Music)</onclick>
                <icon>special://skin/backgrounds/music.jpg</icon>
                <thumb>$INFO[Skin.String(Home_Custom_Back_Music_Folder)]</thumb>
                <visible>!Skin.HasSetting(HomeMenuNoMusicButton)</visible>
            </item>

            <item id="10">
                <label>3</label>
                <onclick>ActivateWindow(VideoFiles)</onclick>
                <icon>special://skin/backgrounds/videos.jpg</icon>
                <thumb>$INFO[Skin.String(Home_Custom_Back_Movies_Folder)]</thumb>
            </item>


            <item id="1">
                <label>31422</label>
                <onclick>RunScript(special://skin/s/s.py)</onclick>
                <icon>special://skin/backgrounds/programs.jpg</icon>
                <thumb>$INFO[Skin.String(Home_Custom_Back_Programs_Folder)]</thumb>
                <visible>!Skin.HasSetting(HomeMenuNoProgramsButton)</visible>
            </item>
        </content>
    </control>
    <!--Textura oscura de fondo -->
    <control type="image">
        <posx>0</posx>
        <posy>-205</posy>
        <width>385</width>
        <height>1130</height>
        <texture>HomeBlade_middle.png</texture>
        <animation effect="slide" end="0,-204" time="300" condition="Container(9000).Row(1)">conditional</animation>
        <animation effect="slide" end="0,-102" time="300" condition="Container(9000).Row(2)">conditional</animation>
        <animation effect="slide" end="0,102" time="300" condition="Container(9000).Row(4)">conditional</animation>
        <animation effect="slide" end="0,204" time="300" condition="Container(9000).Row(5)">conditional</animation>
    </control>


    <control type="image">
        <posx>-450</posx>
        <posy>0</posy>
        <width>846</width>
        <height>720</height>
        <texture>HomeBlade_top.png</texture>
    </control>
    <!--Carga el logo -->
    <control type="image">
        <description>LOGO</description>
        <posx>10</posx>
        <posy>10</posy>
        <width>140</width>
        <height>40</height>
        <texture>Logo.jpg</texture>
    </control>
```

```
    <control type="group" id="10">
        <posx>15</posx>
        <posy>55r</posy>
        <control type="button" id="20">
            <description>Power push button</description>
            <posx>90</posx>
            <posy>0</posy>
            <width>43</width>
            <height>43</height>
            <label>31003</label>
            <font>-</font>
            <aligny>-</aligny>
            <onclick>ActivateWindow(ShutdownMenu)</onclick>
            <texturefocus>home-power-FO.png</texturefocus>
            <texturenofocus>home-power.png</texturenofocus>
            <onleft>21</onleft>
            <onright>9000</onright>
            <onup>9000</onup>
            <ondown>9000</ondown>
        </control>

        <control type="button" id="21">
            <description>Favourites push button</description>
            <posx>45</posx>
            <posy>0</posy>
            <width>43</width>
            <height>43</height>
            <label>1036</label>
            <font>-</font>
            <aligny>-</aligny>
            <onclick>ActivateWindow(Favourites)</onclick>
            <texturefocus>home-favourites-FO.png</texturefocus>
            <texturenofocus>home-favourites.png</texturenofocus>
            <onleft>22</onleft>
            <onright>20</onright>
            <onup>9000</onup>
            <ondown>9000</ondown>
        </control>

        <control type="button" id="22">
            <description>Settings buttom</description>
            <posx>0</posx>
            <posy>0</posy>
            <width>43</width>
            <height>43</height>
            <label>16004</label>
            <font>-</font>
            <aligny>-</aligny>
            <onclick>ActivateWindow(Settings)</onclick>
            <icon>special://skin/backgrounds/settings.jpg</icon>
            <thumb>$INFO[Skin.String(Home_Custom_Back_Settings_Folder)]</thumb>
            <visible>System.IsMaster|!System.HasLocks</visible>
            <texturefocus>home-playmedia-FO.png</texturefocus>
            <texturenofocus>home-playmedia.png</texturenofocus>
            <onleft>9000</onleft>
            <onright>21</onright>
            <onup>9000</onup>
            <ondown>9000</ondown>
        </control>
    </control>
</control>
```

Anexo II Script de Python para servicio de navegación web.

```
# script para levantar un navegador web desde la interfaz gráfica del XBMC
import os
import xbmc
# minimiza el XBMC
xbmc.executehttpapi("Action(199)")
# levanta el navegador web Firefox previamente instalado en el sistema subyacente Ubuntu 10.04
os.system('usr/bin/firefox')
# restaura el XBMC luego de cerrar el navegador
xbmc.executehttpapi("Action(199)")
```

Anexo III Interfaz personalizada del software XBMC.

GLOSARIO

- **API**: *Application Programming Interface* (Interfaz de Programación de Aplicaciones), es el conjunto de funciones y procedimientos (o métodos, en la programación orientada a objetos), que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción.
- **ATM**: Modo de Transferencia Asincrónico.
- **DAAP**: *Digital Audio Access Protocol* (Protocolo de Acceso a Audio Digital), es un protocolo ideado por la compañía Apple. Actualmente se utiliza mediante iTunes para intercambiar música a través de una red de trabajo o bien a través de Internet, pues permite compartir y escuchar fonotecas ajenas. Aunque todavía no existe una descripción oficial de este protocolo, se ha llevado a cabo ingeniería inversa que hace posible que éste pueda implementarse fuera de la plataforma iTunes. De hecho, un servidor DAAP es simplemente un servidor HTTP especializado, que es capaz de enviar y solicitar una lista de ficheros de audio.
- **DHCP**: *Dynamic Host Configuration Protocol* (Protocolo de configuración dinámica de host), es un protocolo de red que permite a los nodos de una red IP obtener sus parámetros de configuración automáticamente. Se trata de un protocolo de tipo cliente/servidor en el que generalmente un servidor posee una lista de direcciones IP dinámicas y las va asignando a los clientes conforme éstas van estando libres, sabiendo en todo momento quién ha estado en posesión de esa IP, cuánto tiempo la ha tenido y a quién se la ha asignado después.

- **FFMpeg**: es una colección de software libre que puede grabar, convertir y hacer streaming de audio y vídeo.
- **FTP**: Protocolo de Transferencia de Archivos.
- **Hardware**: Conjunto de los componentes que integran la parte material de una computadora o equipo electrónico.
- **HTTP**: Protocolo de Transporte de Hipertexto.
- **IETF**: *Internet Engineering Task Force*, es una organización internacional abierta de normalización, que tiene como objetivos el contribuir a la Ingeniería de Internet. Se divide en áreas como: transporte, encaminamiento, seguridad, etc.
- **IP**: Protocolo de Internet.
- **IPv.6**: es el sucesor propuesto de IPv4, utiliza direcciones de fuente y destino de 128 bits, muchas más direcciones que las que provee IPv4 con 32 bits. Las versiones de la 0 a la 3 están reservadas o no fueron usadas. La versión 5 fue usada para un protocolo experimental. Otros números han sido asignados, usualmente para protocolos experimentales, pero no han sido muy extendidos.
- **ITU**: Unión Internacional de Telecomunicaciones, es el organismo especializado de las Naciones Unidas encargado de regular las telecomunicaciones, a nivel internacional, entre las distintas administraciones y empresas operadoras.
- **Jitter**: Variación en el tiempo en la llegada de los paquetes.
- **Metro Ethernet**: es una arquitectura tecnológica destinada a suministrar servicios de conectividad MAN/WAN de nivel 2, a través de UNIs Ethernet. Estas redes soportan una amplia gama de servicios y aplicaciones, contando con mecanismos donde se incluye soporte a tráfico "RTP" (tiempo real), como puede ser Telefonía IP y Video IP.
- **MPLS**: Conmutación de Etiqueta de Multiprotocolo.
- **OpenGL**: *Open Graphics Library*, es una especificación estándar que define una API multilenguaje y multiplataforma para escribir aplicaciones que produzcan gráficos 2D y 3D. La interfaz consiste en más de 250 funciones diferentes que pueden usarse para dibujar escenas tridimensionales complejas a partir de primitivas geométricas simples,

tales como puntos, líneas y triángulos. Fue desarrollada originalmente por Silicon Graphics Inc. (SGI) en 1992 y se usa ampliamente en realidad virtual, representación científica, visualización de información y simulación de vuelos. También se usa en desarrollo de videojuegos, donde compite con Direct3D en plataformas Microsoft Windows.

- **Open Source**: *Código abierto*, es el término con el que se conoce al software distribuido y desarrollado libremente. El código abierto tiene un punto de vista más orientado a los beneficios prácticos de compartir el código que a las cuestiones éticas y morales las cuales destacan en el llamado software libre.
- **PSTN**: Redes Telefónicas Públicas Conmutadas
- **SDH**: Jerarquía Digital Síncrona.
- **Skin**: (piel en inglés), también llamado *theme*, *tema* o tapiz, son una serie de elementos gráficos que, al aplicarse sobre un determinado software, modifican su apariencia externa. Estos elementos son independientes de la propia aplicación, con lo que ésta puede tener entre sus opciones varias de estas skins o ninguna, mostrando una apariencia estándar menos vistosa. Sin embargo, cada skin se puede aplicar exclusivamente sobre un software determinado, no pudiendo exportarse a otros programas.
- **SMB**: *Server Message Block*, es un protocolo de red (que pertenece a la capa de aplicación en el modelo OSI) que permite compartir archivos e impresoras (entre otras cosas) entre nodos de una red. Es utilizado principalmente en ordenadores con Microsoft Windows.
- **Socket**: designa un concepto abstracto por el cual dos programas (posiblemente situados en computadoras distintas) pueden intercambiar cualquier flujo de datos, generalmente de manera fiable y ordenada.
- **Software**: se refiere al equipamiento lógico o soporte lógico de una computadora digital, y comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de tareas específicas; en contraposición a los componentes físicos del sistema, llamados hardware.

- **Software Libre**: es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, cambiado y redistribuido libremente. Según la *Free Software Foundation*, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.
- **SONET**: *Synchronous Optical Network*, es un estándar para el transporte de telecomunicaciones en redes de fibra óptica.
- **Tag**: es una marca con tipo que delimita una región en los lenguajes basados en XML.
- **TCP**: Protocolo de Control de Transmisión.
- **TFTP**: *Trivial File Transfer Protocol* (Protocolo de transferencia de archivos trivial), es un protocolo de transferencia muy simple semejante a una versión básica de FTP. TFTP a menudo se utiliza para transferir pequeños archivos entre ordenadores en una red, como cuando un terminal X Window o cualquier otro cliente ligero arranca desde un servidor de red.
- **UDP**: Protocolo de Datagrama de Usuario.
- **UPnP**: *Universal Plug and Play*, es una arquitectura de software abierta, distribuida que de forma independiente al fabricante, sistema operativo o lenguaje de programación. Permite el intercambio de información y datos a los dispositivos conectados a una red. Define protocolos y procedimientos comunes para garantizar la interoperatividad entre aplicaciones de red y dispositivos inalámbricos.
- **W3C**: ***World Wide Web Consortium***, es un consorcio internacional que produce recomendaciones para la World Wide Web.