UNIVERSIDAD POLITÉCNICA SALESIANA

"DISEÑO DE UNA PLATAFORMA DE SOFTWARE PARA TELEVISIÓN DIGITAL INTERACTIVA DE UN CANAL DE DEPORTES UTILIZANDO GINGA-NCL LUA"

Facultad de Ingenierías

Tesis previa a la obtención del

Título de Ingeniero Electrónico

Dirigido por: Ing. Edgar Ochoa Figueroa

Autor:

Miguel Alberto Dávila Sacoto

CUENCA, ENERO DE 2012

DEDICATORIA

A mi mamá, quién sin su apoyo, dedicación y esfuerzo,

mi formación no hubiese sido posible.

AGRADECIMIENTO

A Dios, a mis amigos, y a mi familia.

Ing. Edgar Ochoa Figueroa

CERTIFICA

Haber dirigido y revisado prolijamente cada uno de los capítulos de este trabajo de tesis, realizado por el señor Miguel Alberto Dávila Sacoto.

Por cumplir los requisitos, autorizo su presentación.

_____

Ing. Edgar Ochoa Figueroa

DECLARATORIA DE RESPONSABILIDAD

Yo, Miguel Alberto Dávila Sacoto, autor del presente Trabajo de Tesis titulado “DISEÑO DE UNA PLATAFORMA DE SOFTWARE PARA TELEVISIÓN DIGITAL INTERACTIVA DE UN CANAL DE DEPORTES UTILIZANDO GINGA-NCL LUA” declaro que:

Los conceptos desarrollados, análisis realizados y las conclusiones del presente trabajo, son de exclusiva responsabilidad del autor.

Cuenca, enero de 2012

_____

Miguel Alberto Dávila Sacoto

LOS MIEMBROS DEL TRIBUNAL DE CALIFICACIÓN Y SUSTENTACIÓN DE LA TESIS

"DISEÑO DE UNA PLATAFORMA DE SOFTWARE PARA TELEVISIÓN DIGITAL INTERACTIVA DE UN CANAL DE DEPORTES UTILIZANDO GINGA-NCL LUA"

Que la presente tesis ha sido debidamente revisada y por lo tanto

APROBADA

_____

Presidente Del Tribunal

_____

Miembro Del Tribunal

_____

Miembro Del Tribunal

Cuenca, enero de 2012

Prefacio

El desarrollo de aplicaciones para televisión digital es un campo poco explotado en nuestro país. Además la documentación existente está únicamente en portugués, por lo cual no se tiene un adelanto muy importante en este campo, lo cual hace emergente una guía de configuración y programación en los lenguajes que tienen lugar este tipo de aplicaciones.

Existen pocos documentos redactados del tema en nuestro país, quizá uno de los más importantes y que se tomó como referencia para el presente trabajo, es la tesis de Javier Eduardo Torres Altamirano de la Escuela Politécnica del Ejército con el tema "Diseño y desarrollo de una aplicación de contenidos interactivos para TV Digital basada en el Middleware Ginga del sistema Brasileño", en la cual se da una guía de programación en NCL basada en los documentos de la Puc-Río. Además en cuanto a Hardware existe la tesis de Fernando Muñoz Fernándes de Legaria de la Universidad Pública de Navarra con el tema "Implementación de una solución para la gestión de un canal de televisión digital terrestre", que aunque está basado en la norma europea, consiste en un buen referente para la selección de hardware y software para televisión digital. Estas publicaciones, sumadas a los esfuerzos por el laboratorio LIFIA de argentina constituyen las principales referencias en español para el desarrollo de aplicaciones, por lo cual se requiere seguir trabajando en el tema para lograr una base de información para los desarrolladores.

Por lo tanto, el presente trabajo abarca el trabajo realizado en los documentos ya descritos, conjuntamente con un análisis a nivel local, lo cual nos permitirá tener una noción más acertada sobre la realidad de la televisión digital en nuestro país, y pretende constituir una base para el desarrollo de aplicaciones dentro de la Universidad Politécnica Salesiana sede Cuenca.

Prólogo

En el presente trabajo se realizará un breve resumen sobre la norma brasileña para televisión digital terrestre, permitiendo al lector una noción sobre los requisitos necesarios para diseñar e implementar una plataforma de software y hardware de un canal de televisión digital.

Se especificarán los paquetes de software necesarios para el diseño de contenidos interactivos, pasando desde la instalación de las máquinas virtuales y culminando en la codificación del diagrama NCM al lenguaje NCL y Lua. Con esto se pretende lograr una mejor aproximación sobre el tema, ya que la información que se encuentra normalmente sobre el mismo aún sigue siendo escasa.

Además se realizará una comparación sobre soluciones de software propietario y software libre para los procesos de multiplexación y decodificación, así también como la cotización pertinente, con lo que se puede tener un referente para la implementación de una solución integral para una empresa de este tipo.

Índice General:

Introducción

En el año 2010, el Ecuador adoptó la norma Japonesa con modificaciones Brasileñas (SBTVD) de televisión digital para realizar el cambio requerido para liberar el espectro de frecuencias utilizado por la televisión analógica. Este proceso viene acompañado de un cambio en la plataforma de transmisión, y en los equipos de recepción del usuario, además dentro de los equipos destinados a la transmisión y a la gestión de contenidos, la manera de empaquetar video, voz y datos se realiza utilizando el middleware abierto Ginga.

Ginga es el nombre del Middleware Abierto del Sistema Brasileño de TV Digital, este compilador está formado por un conjunto de tecnologías estandarizadas e innovaciones brasileñas que lo convierten en la especificación de middleware más avanzada y la mejor solución para la generación de contenidos de televisión digital para el sistema brasileño y para todos los países que adoptaron esta norma.

El middleware abierto Ginga se subdivide en dos subsistemas principales que permiten el desarrollo de aplicaciones siguiendo dos tipos de programación diferentes. Estos dos subsistemas se llaman Ginga-J (para aplicaciones procedurales Java) y Ginga-NCL (para aplicaciones declarativas NCL). Además existe una modificación de Ginga-C++ pero este tipo de programación aún no es aceptado mundialmente y se encuentra en desarrollo.

Ginga es el fruto del desarrollo de proyectos de investigación, liderados principalmente por empresas brasileñas, coordinados por los laboratorios Telemidia de la PUC-Rio y LAViD de la UFPB.

Debido al cambio de televisión analógica a televisión digital que se dará en nuestro país en el año 2017, el desarrollo de software para la generación de contenidos es una necesidad inminente, por lo que la aplicación el lenguaje descriptivo NCL LUA es igual de imperativo. El uso de plataformas de gestión de software en las empresas que se dedican a transmitir canales de televisión se totalizará luego del apagón analógico, lo cual demandará profesionales en la materia.

Una aplicación muy puntual es la que se estudiará en esta tesis, la cual corresponde a la gestión de archivos multimedia y de contenido digital para un canal de deportes abierto y un canal de deportes pagado, cada uno con sus respectivos requerimientos. Con esto se dará una guía para la generación de contenidos utilizando Ginga-NCL, y se analizará la manera de aplicar las herramientas del lenguaje descriptivo NCL LUA a una situación real, manejo de contenidos en tiempo real, gestión de archivos multimedia y gestión de usuarios al utilizar un canal de retorno.

El problema que se pretende solventar con esta tesis es el uso de Ginga-NCL para una aplicación real, es decir, aplicar todos los conocimientos sobre el estándar Japonés y brasileño, así como los manuales de uso de Ginga, y converger en una solución para un caso de un canal de deportes abierto y pagado.

En el caso de un canal abierto, se requiere gestionar el contenido multimedia, y el contenido digital que será transmitido al usuario final. En este caso, no se tiene un canal de retorno, así, el usuario es un espectador del contenido que le ofrece el canal de televisión,

pero puede interactuar con el contenido de un programa en especial, en el caso de un canal de deportes: formación de equipos en un partido, tablas de posiciones, información de locutores, etc.

En el caso de un canal pagado, se requiere gestionar cuentas de usuarios para el canal de retorno, así el usuario final ya no es sólo un espectador, sino puede interactuar directamente con los programas en vivo, tal es el caso de concursos, foros, etc., todo mediante un canal de retorno utilizando internet.

Capítulo I:

DESCRIPCIÓN DE LA NORMA SBTVD-T PARA TELEVISIÓN DIGITAL Y SUS MODIFICACIONES A LA NORMA ISDB-T

1.1 – Introducción

Cuando se hace referencia a la televisión digital, se engloban tecnologías de transmisión y recepción, canales de retorno, y producción de contenidos, así también como la posibilidad de transmitir varias señales dentro de un mismo canal.

La televisión digital terrestre o TDT por sus siglas en español, es la aplicación de estas tecnologías digitales a la transmisión y recepción de contenidos en lo que normalmente conocemos como televisión analógica. Esta opción ofrece grandes ventajas como la transmisión de canales en alta definición, mejorar la calidad de recepción, guía digital de programación, servicios interactivos, etc., lo cual cambia por completo la manera en la que se ve televisión actualmente.

1.2 - Historia de la Televisión Digital Terrestre

La televisión digital terrestre, nación como un intento de transmitir canales de televisión en alta definición (HDTV). HDTV fue creada por la NHK (Corporación Emisora de Japón, o Asociación de Radiodifusión de Japón), cuya investigación empezó en 1960, y lo presentó como estándar a la ITU en 1973.

Luego de los estudios sobre alta definición en 1980 y los avances sobre televisión satelital, la Comisión Federal de Comunicaciones (FCC), junto con las principales cadenas de televisión, fabricantes, y universidades, crearon un estándar digital para la transmisión de televisión digital terrestre, la cual permita una compatibilidad con los televisores analógicos existentes. Este nuevo sistema dejaría libres la banda de 470 a 890Hz, así como las frecuencias UHF, y VHF, para luego ser reasignadas a los canales de televisión digital.

Figura 1.1. Logotipo de la Televisión en Alta Definición

Debido a que una imagen en alta definición requería mucho espacio, y un ancho de banda considerable para la transmisión, en 1982, la NHK desarrolló un sistema de compresión y transmisión, el MUSE (codificación múltiple de muestreo sub-nyquist).

En 1990, la FCC creó la ACATS, o comisión asesora sobre servicios de televisión avanzada, la cual se encargaría de la preparación de un estándar técnico para la difusión de canales digitales, debido a que para transmitir una señal de alta definición (HDTV) por un canal NTSC, se tenía un sistema poco eficaz, y sobre todo poco rentable, por lo cual se requería una manera de transmitir la señal HDTV (que requiere un ancho de banda de 6MHz) utilizando las nuevas tecnologías digitales conocidas hasta la fecha.

En 1993 se constituyó la Gran Alianza, entre AT&T, General Instruments, Philips, Thomson Electronics, Zenith, el centro de investigación de David Sarnoff, y el Tecnológico de Massachussets. Estas empresas trabajaron en conjunto para introducir mejoras a los sistemas analizados por la ACATS sobre todo al sistema de MUSE, con lo cual trabajaron en aspectos como la compresión de datos de audio y video, técnicas de multiplexación para los mismos, y técnicas de modulación.

Figura1.2. Modelo de aplicación ejecutándose en un terminal de Televisión Digital.

Con estos avances se dejó de llamar a estos sistemas como televisión de Alta Definición, y junto con la transmisión de datos, se lo llamó Televisión Digital Terrestre, así , en 1995 nació la norma ATSC (Advanced Television Standard), utilizada en Estados Unidos, Canadá y México. Esta decisión de Estados Unidos, obligó a los japoneses a modificar su estándar, con lo cual en 1999 nació el sistema ISDB-T (Integrated Services Digital Broadcasting-Terrestrial).

El estándar para televisión digital terrestre utilizado en la mayor parte de Latinoamérica es el sistema Brasileño, el cual se conoce como SBTVD (en portugués, Sistema Brasileiro de Televisão Digital) o ISDB-Tb (Integrated Services Digital Broadcasting Terrestrial Built-in, por sus siglas en inglés). Dicho estándar es una modificación del sistema de televisión digital Japonés ISDB-T, esta modificación se basa en aspectos técnicos en cuanto a la compresión de video y audio, el software para la interacción con el usuario y la velocidad de modulación.

Para hacer un análisis entre estas dos normas, es necesario tomar en cuenta los siguientes aspectos técnicos:

- Modulación utilizada
- Codificación de error
- Multiplexación
- Transmisión de datos
- Codificación de Audio
- Codificación de Video

1.3 - Características Generales de la norma ISDB-T

ISDB-T es la norma japonesa para la transmisión de televisión digital terrestre, la cual además de transmitir audio y video, permite al generador de contenidos transmitir datos por el canal, lo cual permite al usuario final interactuar con el canal de televisión.

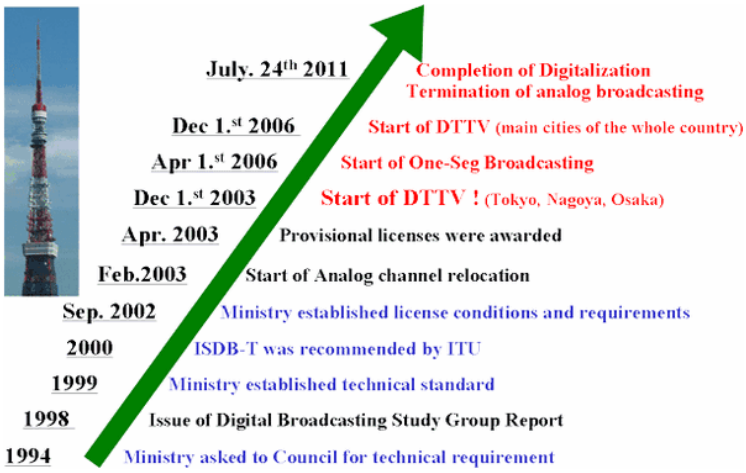


Figura 1.3. Cronología de la evolución de la Televisión Digital en Japón [14]

Esta norma, permite también, transmitir varios canales de datos y multimedia sobre la misma banda, lo cual optimiza el uso del espectro radioeléctrico, lo cual se logra mediante SFN(Single Frequency Network, o red de frecuencia única, mediante la cual los equipos de transmisión, emiten la señal en el mismo canal de frecuencia), y sobre todo entrega al usuario final la capacidad de escoger el entre varios programas (o subprogramas) de una misma cadena televisiva.

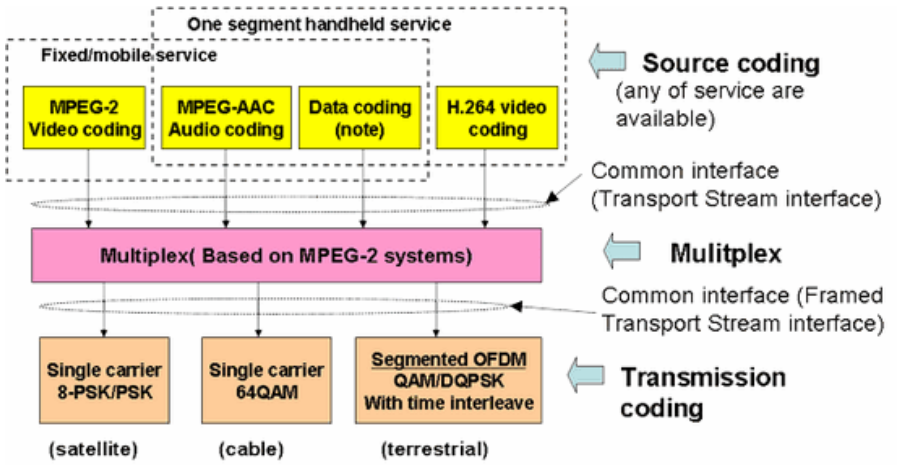


Figura 1.4. Modelo de transmisión utilizado en la norma Japonesa [14]

Así, ISDB-T en cuanto a los canales de audio y video, permite la transmisión de un canal en alta definición (HDTV) en un ancho de banda de 6 MHz, o se lo puede subdividir entre dos y tres canales de definición estándar (SDTV). En la figura se muestra la segmentación del canal de transmisión digital y las tres configuraciones posibles.

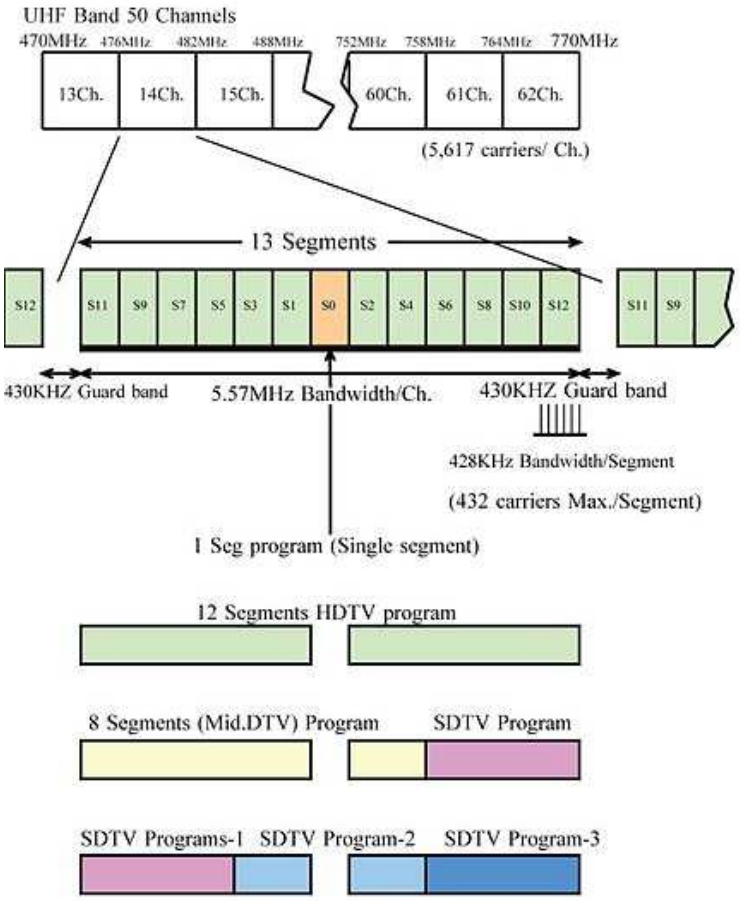


Figura 1.5. Segmentación de un canal ISDB [1]

ISDB-T permite el envío de datos, y servicios interactivos con el usuario final, utilizando como Internet como canal de retorno. Esta característica es completamente innovadora frente a la televisión analógica, pues se puede proporcionar al usuario la guía electrónica de programas (o EPG por sus siglas en inglés de Electronic Program Guide), servicios de compras en línea, juegos, etc. Esto abre un mundo nuevo de posibilidades, donde el rol del televidente cambia por completo, y lo convierte en un ser activo frente a un televisor.

Otro factor importante de ISDB-T, es las ventajas que presenta ante la interferencia, siendo muy robusta ante multitrayectoria, y al ruido de señales UHF (presente en líneas de transmisión y motores de inducción).

1.3.1 -Tipo de compresión y multiplexación

ISDB-T utiliza MPEG2 para la compresión y multiplexación de audio y video, lo cual permite transmitir imágenes en alta definición, para esto se requiere una elevada

tasa de bits de transmisión, por lo que no es recomendable en redes móviles de recursos limitados.

1.3.2 Modulación

ISDB en general utiliza 64QAM-OFDM, 16QAM-OFDM, QPSK-OFDM, y DQPSK-OFDM, dependiendo de los requerimientos de la banda de frecuencia. Para televisión digital se utiliza PSK y QAM con multiplexación COFDM (multiplexación por división de frecuencias ortogonales).

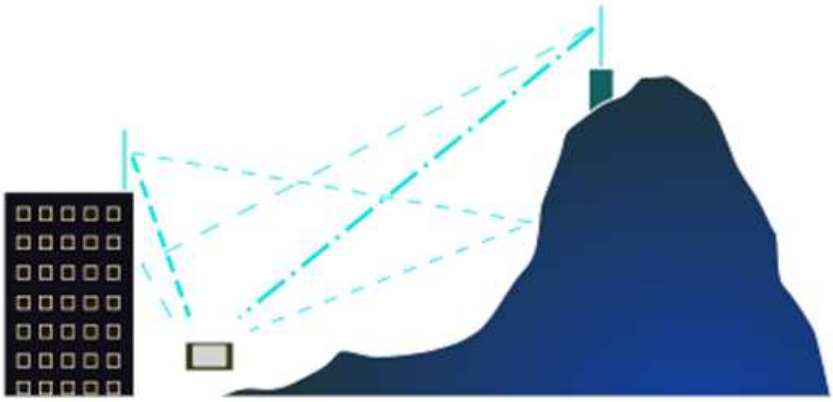


Figura 1.6: COFDM: La duración de bits es superior a los retardos, evitando ecos y permitiendo reutilizar frecuencias [14]

1.4 Características Generales de la norma SBTVD-T

SBTVD –T es la norma brasileña para la transmisión de televisión digital terrestre, la cual es una modificación a la norma japonesa ISDB-T para acoplarse de mejor manera al mercado brasileño, y por extensión, al mercado latinoamericano.

El factor más importante que diferencia a ambas normas es el uso del middleware Ginga en el sistema brasileño. El middleware Ginga, es un software de soporte que permite el intercambio, y gestión de aplicaciones de una manera más fácil para los desarrolladores. Ginga es basado en NCL (Nested context Language por sus siglas en inglés), mientras que ISDB-T se basa en BML (Broadcast Markup Language).

En general SBTVD es la norma más moderna, y tecnológicamente más avanzada, por lo que constituye una alternativa excelente para llevar adelante procesos de investigación. Además es de licencia libre, por lo que los costos de la red y para el

consumidor final disminuyen. También es una norma de mucha explotación en términos económicos para los proveedores de servicio, pues al basarse en la norma japonesa, y aumentar la compresión de video, se pueden enviar una mayor cantidad de canales, variando su calidad, y permitiendo su transmisión a equipos móviles.

1.4.1 Tipo de compresión y multiplexación

Una modificación es el uso de la codificación H.264 o más conocida como MPEG-4 para la compresión de video en la norma brasileña, mientras que la norma nipona utiliza un sistema de compresión en MPEG-2. Esta característica es trascendental, pues MPEG-4 fue diseñado para la transmisión de contenido web para dispositivos con recursos limitados y conexiones de internet con tasas de bit bajas, lo cual se adapta de mejor manera a las circunstancias propias de los países latinoamericanos. Además, SBTVD-T puede transmitirse en 6 MHz, 7 MHz y 8MHz, siendo completamente compatible con ISDB-T.

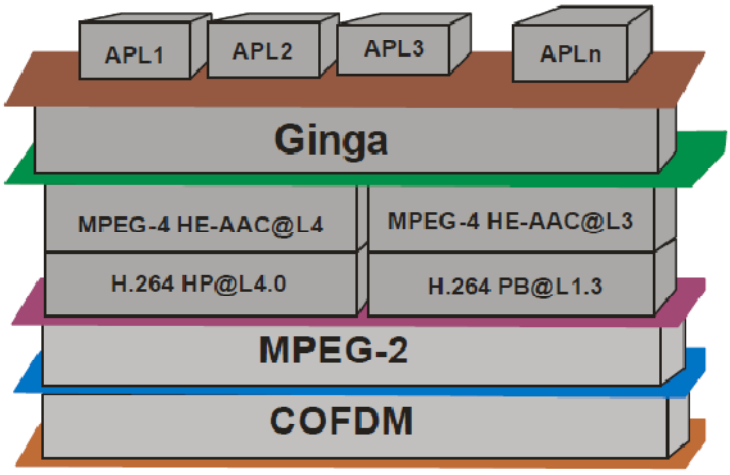


Figura 1.7. Capas de multiplexación y compresión del sistema brasileño [1]

1.4.2 Modulación

SBTVD-T utiliza BST-OFDM (Band Segmented Transmission Orthogonal Frecuency Division Multiplexing por sus siglas en inglés) en el cual se divide la banda de frecuencia de un canal en trece segmentos como en ISDB, pero además se puede cambiar el esquema del segmento para la emisión del canal.

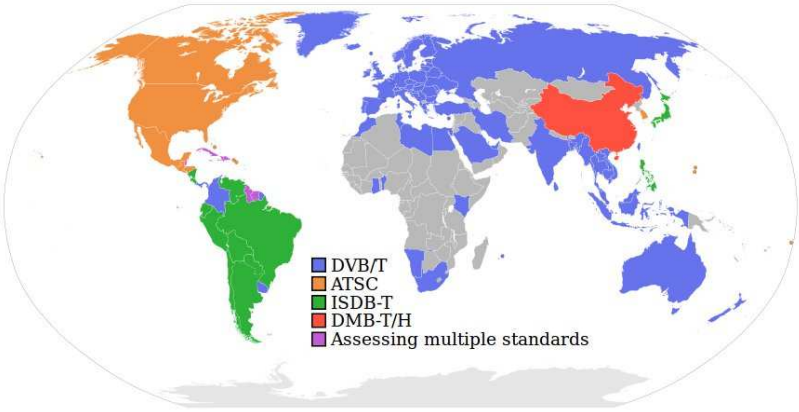


Figura 1.8. Zonas de utilización de las normas para televisión digital [13]

1.5 Infraestructura de una red para televisión digital

1.5.1 Infraestructura física

La infraestructura de un sistema de televisión digital posee como mínimo tres componentes; dos en la infraestructura del proveedor: el servidor de aplicaciones y contenidos y el servidor de playout; y uno en la infraestructura del usuario: el set top box. Dichos elementos se deben seleccionar de acuerdo a los requerimientos propios de la empresa que brindará este servicio, y a la naturaleza del tipo de aplicación.

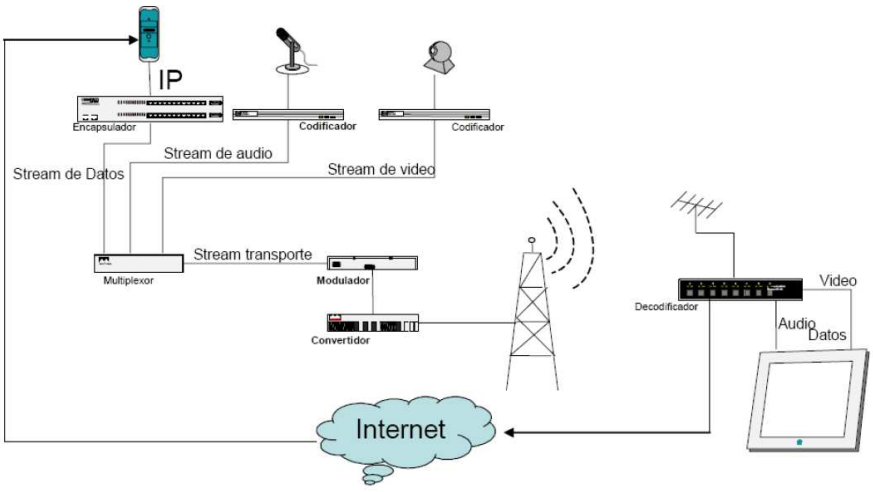


Figura 1.9. Infraestructura básica de una red de televisión digital [3]

En la figura 1.9 se muestra la infraestructura física de una red de televisión digital. Los streamings de audio, video y datos se multiplexan en un stream de transporte MPEG-2, el cual se modula a través de una placa física. La señal se captura con una antena, se demodula y decodifica en el STB, el cual mediante el middleware que utilice, se encarga de reproducir el contenido del stream.

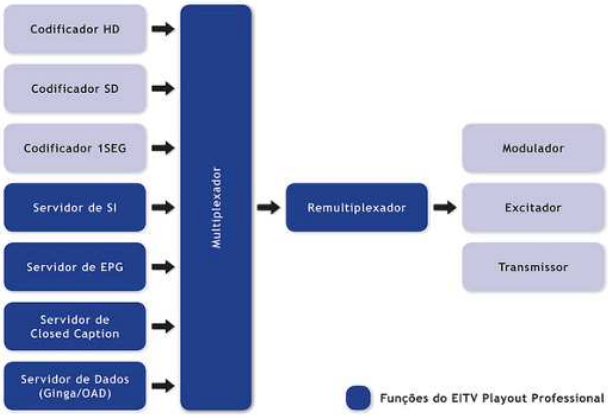
1.5.1.1 El servidor de aplicaciones y contenidos

Un servidor de aplicaciones consiste en un equipo que se encarga de almacenar y distribuir las aplicaciones NCL que se ejecutarán en la transmisión. Además debe comunicarse, o albergar en su interior software para el desarrollo de dichas aplicaciones.

1.5.1.2 El servidor de Playout

El servidor de PlayOut, es un equipo encargado de la difusión y modulación de la señal de televisión. Este equipo genera el stream (flujo de datos) de transporte, utilizando como norma MPEG-2 (norma ISDB).

(a)



(b)

Figura 1.10. Equipo comercial de PlayOut: (a) Equipo físico (b) Funciones de PlayOut profesional

Este equipo es un multiplexor, que se encarga de unir la aplicación NCL con el flujo de audio y video de un servidor para este caso, además de información que demanda el servidor Ginga como el sistema de archivos para el carrousel de objetos (encargado de la actualización de aplicaciones en el set top box), y los canales propios de mantenimiento y emergencia pertinentes en la red. Este equipo es un conjunto de hardware muy robusto, pues demanda un procesamiento elevado, a altas tasas de transmisión, y requiere funcionar continuamente.

1.5.1.3 El Set Top Box (STB) o equipo de usuario

Un STB consiste en un equipo que se encarga de recibir la señal de la antena del receptor, decodificarla y mostrarla en un televisor. En este equipo corre el middleware, Ginga en el caso de ISDBTb, el cual ejecuta la aplicación de interactividad.

Figura 1.11. Set Top Box comercial.

Es un equipo tecnológicamente muy avanzado y veloz, pues utiliza microcontroladores de gama alta o FPGAs de alta velocidad para el tratamiento de imagen, y aplicaciones. Generalmente poseen un microcontrolador ARM con un kernel de Linux embebido. Posee puertos de entrada y salida de cable coaxial, puertos HDMI, un host USB para cargar aplicaciones o setear parámetros del kernel, y un puerto Ethernet para el canal de retorno.

En Latinoamérica los únicos productores de receptores se encuentran en el mercado Argentino, la Unión Temporaria de empresas (UTE) en conjunto con el laboratorio argentino de televisión digital (NeoTVLab) y el Lifia.

Figura 1.12. Receptor digital UTE

Los receptores digitales de UTE tienen modificaciones en el middleware. Estas modificaciones radican en aspectos como el manejo del carrousel de objetos, y la decodificación del audio. Este conjunto de complementos recibe el nombre de Lifia, el cual es una modificación de Ginga adaptado a la realidad de los usuarios Argentinos.

Además existen otras opciones para receptores de televisión digital, los cuales pueden estar directamente construidos dentro del televisor, o dentro de teléfonos móviles (One-Seg), con lo que el usuario puede acceder e interactuar con los contenidos utilizando su plan de datos. También existen receptores USB para computadores portátiles, permitiendo que los servicios de televisión digital sean accesibles por la mayor cantidad de usuarios.

(a)

(b)

Figura 1.13. Receptores alternos para televisión digital (a)Dispositivos móviles utilizando tecnología One-Seg (b) Dongles USB para recepción en equipos portátiles

1.5.2 Infraestructura de Software para televisión digital

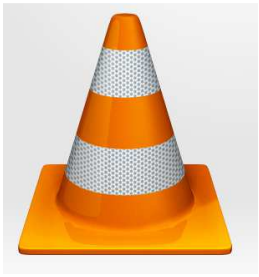
Junto con la infraestructura física, es necesario un conjunto de software que se encargue del control, ejecución y administración de los elementos de hardware. Los requerimientos mínimos para un canal de televisión son los siguientes:

1.5.2.1 Software para el servidor de aplicaciones y contenidos

El software que se ejecuta dentro del servidor de aplicaciones, se encarga del almacenamiento, difusión y servicio de las aplicaciones NCL. Este programa requiere realizar actividades como:

- Empaquetamiento de aplicaciones NCL
- Gestión del canal de difusión de aplicaciones (bajo demanda)
- Gestión de peticiones, almacenamiento y respuesta en el canal de retorno
- Desarrollo de aplicaciones
- Gestión de contenido multimedia

Una solución de software libre para un servidor de contenidos es el uso de VLC Media Player de la empresa VideoLan Organization. Este software permite emitir un stream de video a una dirección IP o por telnet a otro servidor, además mediante líneas de comando permite codificar y decodificar audio y video en múltiples formatos.



(a)

Default	Qt4	Cocoa	Qt4	Native	Qt4
Qt4	✓	◒	✓	-	?
Skins	✓	✗	✓	✗	?
Web	✓	✓	✓	✓	✓
Telnet	✓	✓	✓	✓	✓
Command line	✓	✓	✓	✓	✓
ncurses	✗	✓	✓	?	?
Infrared	✗	✗	✓	✗	✗
Mouse Gestures	✓	✓	✓	✓	✓

(b)

Figura 1.14. VLC Media Player como una solución de software libre para un servidor de contenidos (a) Logotipo de VLC Media Player (b) Características de salida de VLC

El flujo de transporte generado por estos tipos de programas debe cumplir con la norma ISDB, por lo que deben permitir transmitir más de un programa simultáneamente. En el diagrama mostrado en la figura 1.15 se muestra un flujo de transporte común en ISDB, en el cual se puede observar tres bloques principales, el Transport Stream (TS) MPEG2, el canal de información ISDB-TB, y los parámetros de transmisión TMCC.

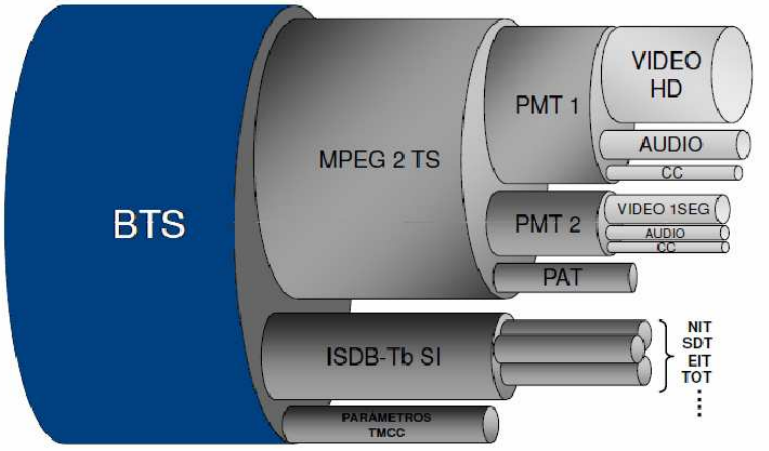


Figura 1.15. Flujo de transporte

1.5.2.2 Software para el servidor de PlayOut

Este software se encarga de la codificación de los contenidos, la multiplexación, y la generación del carrousel de objetos. La generación del carrousel de objetos se basa en el funcionamiento de una máquina de estados, se encuentra especificado en la norma brasileña y su funcionamiento se muestra en la figura 1.16.

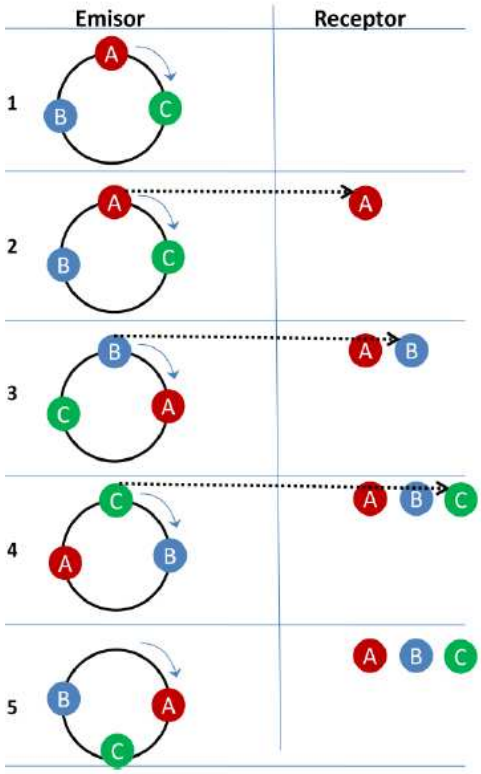


Figura 1.16. Transferencia de archivos a través del carrousel de objetos [2]

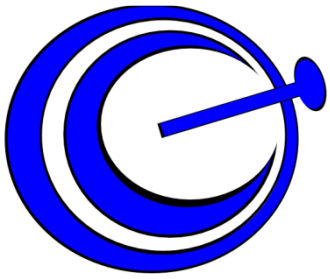
Donde

- Estado 1.- Los receptores no reciben ningún dato en su memoria, lo cual sucede cuando el receptor está apagado. Esta característica se debe tener en cuenta, pues en el sistema brasileño los receptores solicitan las aplicaciones y contenidos al generador del carrousel.
- Estado 2.- Al girar el carrusel, el objeto A se transfiere al receptor.
- Estado 3.- El receptor lee su memoria y si no posee ningún archivo con el identificador de A, lo registrará, y almacenará hasta la carga de todos los archivos. Se procede con la comprobación del archivo B.

- Estado 4.- Se repite el procedimiento de comprobación de memoria y su posterior registro y escritura hasta que se transfiera el último archivo de la aplicación.
- Estado 5.- Una vez que se lea el último identificativo del archivo, la aplicación está lista para ejecutarse.

La solución de software libre que cumple con estas características de generación de carrousel y de multiplexación de contenidos es OpenCaster de la empresa Avalpa Digital Engineering.

(a)



(b)

Figura 1.17. OpenCaster: Solución de software libre para un servidor de PlayOut. (a) Logotipo de la empresa Avalpa Digital Engineering (b) Logotipo del paquete de software OpenCaster

1.6 Tipos de aplicaciones en televisión digital [8]

La televisión interactiva brasileña, soporta varios tipos de servicios para el usuario final, los cuales se clasifican dependiendo de la interactividad que se tiene con la aplicación. Los principales servicios interactivos se agrupan en las siguientes categorías:

1.6.1 t-commerce o aplicaciones de comercio electrónico

Consisten en aplicaciones que aprovechan la publicidad en la televisión, permitiendo que el usuario final realice compras, o interactúe con propagandas de un producto específico. Esta actividad es netamente lucrativa, abriendo un nuevo concepto en el uso del televisor. Dentro de esta categoría, también se incorporan los servicios bancarios, permitiendo al usuario realizar pagos, o transferencias con su código de cuenta secreto.

Figura 1.18. Servicio de T-commerce en TV Digital [13]

1.6.2 t-government o aplicaciones gubernamentales

Estas aplicaciones actualmente presentan un desarrollo muy importante en países como Argentina y Brasil. Aprovechan el potencial de la televisión para llegar a la población de diferentes clases sociales, y agregan la posibilidad de realizar encuestas a los televidentes, o informar de manera ágil e interactiva sobre los procesos que lleva a cabo el gobierno de turno. Este tipo de aplicaciones tienen una connotación especial dentro de la televisión digital, pues dependiendo de las normas del país, todas las empresas de televisión deben tener un canal dedicado para que el gobierno pueda realizar sus informes, consultas, o tratar emergencias nacionales en el momento que se requiera.

1.6.3 t-healt o aplicaciones médicas o de servicio de salud

Consiste en aplicaciones que tienen la misión de ayudar al televidente en temas médicos, como alimentación, ejercicio, etc. No son sistemas inteligentes capaces de recetar o peor aún sustituir a un médico especialista, sino su objetivo es ayudar al televidente a llevar una vida más saludable. En esta clasificación se encuentran aplicaciones como el cálculo del peso ideal, guías de alimentación, etc.

Figura 1.19. Aplicación Viva Mais: T-healt [13]

1.6.4 t-learning o aplicaciones educativas

Este tipo de aplicaciones ofrecen al usuario final una alternativa de educación para niños de todas las edades. Se basan en juegos interactivos con contenido educativo, los cuales ayudan a la formación intelectual de la persona que los utilice.

Figura 1.20. Aplicación de T-learning: SOS Teacher [13]

1.7 El Middleware Ginga

Figura 1.201 Logotipo de la comunidad Ginga [13]

1.7.1Definición de Middleware

Un middleware es una plataforma de software o API (Application Programming Interface), que se implementa dentro de los Set Top Box (STB), la cual permite ejecutar las aplicaciones de televisión interactiva. El Middleware se encarga de la ejecución de las aplicaciones en pantalla, sin la necesidad que el programador tenga un conocimiento profundo sobre la capa de hardware en la que correrá su aplicación.

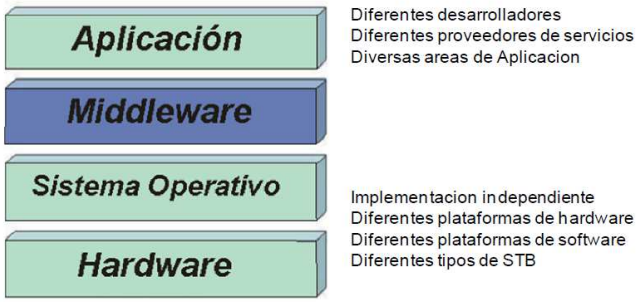


Figura 1.22. Modelo de ejecución de TV digital en un STB [1]

El Middleware Ginga es el utilizado por el sistema brasileño, pero en el mercado existen varios middlewares los cuales se describen brevemente a continuación:

1.7.1.1 Multimedia Home Platform o MHP

Middleware utilizado en la norma europea DVB (Digital Video Broadcasting), es desarrollado sobre Java, basado en una máquina virtual que se encarga de ejecutar la programación en lenguaje assembler del procesador del STB.

Figura 1.23. Logotipo de Multimedia Home Platform

Las aplicaciones en MHP se conocen como DVB-J (por el uso de Java) o Xlet que son independientes del hardware del STB. Tienen el soporte de Sun Microsystems, lo cual lo convierte en una propuesta excelente para desarrolladores de código abierto en cuanto a aplicaciones se refiere. Una API denominada Gestor de Aplicaciones, coordina la comunicación entre la aplicación y el entorno de hardware. El diagrama de bloques de un STB y el Middleware MHP se muestra en la figura 1.24. Como se puede observar en la misma, la API MHP aísla completamente la capa de hardware con la aplicación, lo cual es un factor importante y muy común en los Middlewares de televisión digital, para así fomentar el desarrollo de aplicaciones por parte de personas que no necesariamente conozcan el funcionamiento interno de un STB.

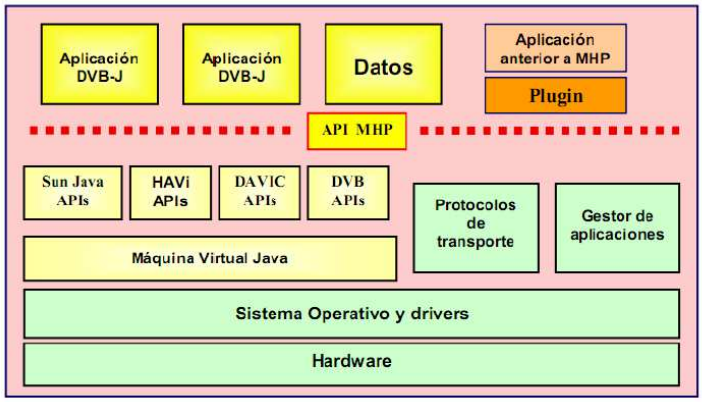


Figura 1.24. Estructura de un STB de norma DVB [1]

1.7.1.2 DASE (Digital Television Application Software Environment)

Es el middleware líder de la ATSC (Advanced Television Systems Committee), adoptado en 2006. La principal aportación de este middleware es que permite al desarrollador, además de escoger la plataforma de software, escoger entre el sistema operativo que utilizará. DASE también es basado en una máquina virtual de Java, por lo que su estructura es muy similar a la de MHP, con la única diferencia es que el sistema operativo es diferente por lo tanto, la interfaz de API's no suele ser compatible. Esta incompatibilidad se soluciona con el uso de GEM (Globally Executable MHP), que es una derivación de MHP, el cual permite que las API's se compartan en nivel de capa de datos.

Las aplicaciones realizadas para DASE permiten su desarrollo utilizando lenguajes declarativos y de procedimiento, ya que se basa en XHTML, permitiendo al desarrollador de páginas web tener un amplio campo en la programación de contenido de TV digital.

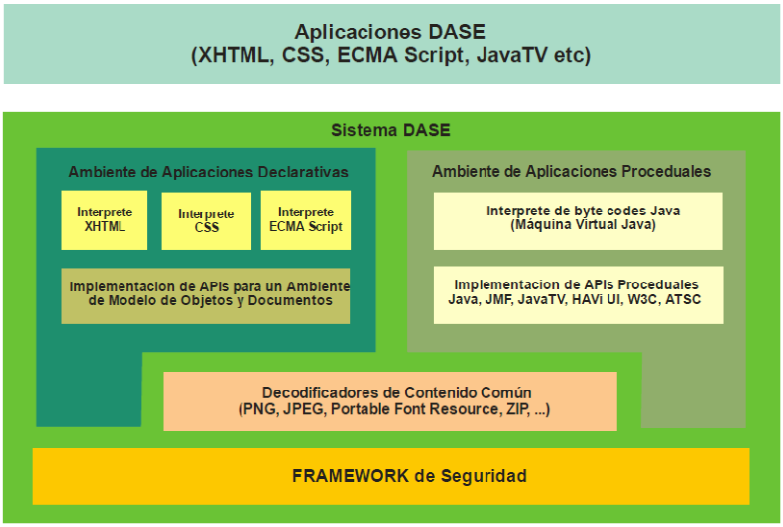


Figura 1.25. Arquitectura del middleware DASE [1]

En la figura 1.25 se muestra la arquitectura del middleware DASE, en la cual se puede observar los módulos principales de aplicación: el declarativo (DAE o Declarative Application Environment) y el procedimental (PAE o Procedural Application Environment), involucra un Framework de script para seguridad en la comunicación, y el Declarative Content Decoding Engine o DCDE que se encarga de la demultiplexación, y decodificación del contenido.

1.7.1.3 ARIB (Association of Radio Industries and Businesses)

Figura 1.26. Logotipo de ARIB

Middleware del estándar japonés ISDB, adoptado en 1997. Define dos estándares de regulación y control en televisión digital: la norma ARIB STD-B23, que se basa en la ejecución de las aplicaciones y servicios basado en MHP; y la norma ARIB STD-23, que especifica el proceso de codificación y transmisión en lenguaje BML (Broadcast Markup Language).

Como se basa en MHP, su ejecución también utiliza una máquina virtual de Java lo que brinda independencia del hardware así las aplicaciones y contenidos BML especifican el comportamiento temporal y espacial del video a transmitir. En la figura 1.27 se muestra el diagrama de bloques de la arquitectura ARIB.

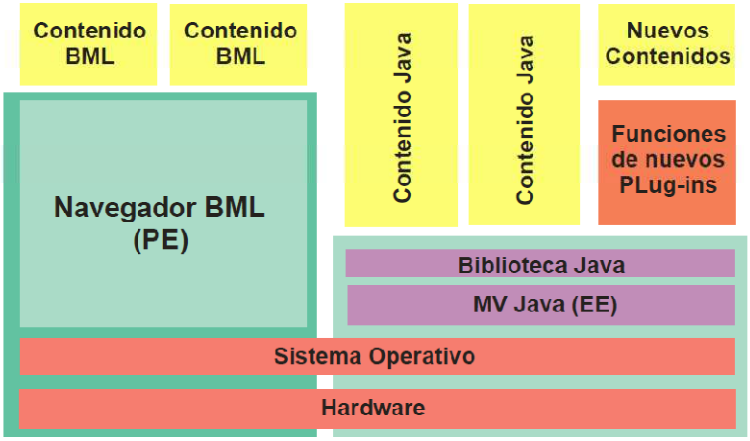


Figura 1.27. Arquitectura del middleware ARIB [1]

1.7.2 Introducción

Ginga es el middleware abierto utilizado en el sistema brasileño SBTVD, el cual es considerado el middleware más avanzado tecnológicamente. Es un middleware

basado en Linux y MHP, que permite la ejecución del lenguaje declarativo NCL (Nested Context Language) y del lenguaje descriptivo Lua.

Al ser una alternativa de código abierto, ha ganado gran popularidad e interés por parte de desarrolladores alrededor del mundo. El middleware fue desarrollado íntegramente en Brasil, con el trabajo conjunto de ingenieros de la PUC-Río (Pontificia Universidad Católica de Río de Janeiro) así como el lenguaje de Scripting Lua.

Ginga está subdividido en Ginga-J para aplicaciones en Java, y Ginga-NCL para aplicaciones declarativas utilizando NCL-Lua. Ambos tipos de programación son completamente diferentes, el primero, Ginga-J, ofrece una alternativa para los desarrolladores de Java, permitiendo la migración de aplicaciones sin importar la plataforma sobre la cual se ejecutará (Linux o sistemas operativos pagados), mientras que NCL permite una herramienta de software libre para el desarrollo de aplicaciones interactivas.

Figura 1.28. Logotipo de la comunicad Ginga-NCL [13]

Se debe tener en cuenta que Ginga-J aún está en proceso de desarrollo, por lo cual no es soportado por la mayoría de Set Top Boxes que se venden en Latinoamérica. De igual manera, la máquina virtual para depuración de Ginga, que ofrecen los Laboratorios Telemidia y la PUC-Rio no es compatible con Ginga-J, por lo cual la cantidad de información es limitada, mientras que NCL, al llevar varios años de desarrollo, tiene herramientas más completas y es soportado por todos los STB's comerciales.

1.7.3 Arquitectura

Ginga, como se mencionó anteriormente, se basa en MHP con modificaciones. Una de las principales es el uso de lenguajes declarativos como NCL, que permiten de manera intuitiva el desarrollo de una aplicación. Ginga se divide en módulos

basados en la programación y el uso del middleware, como se observa en la figura 1.29.

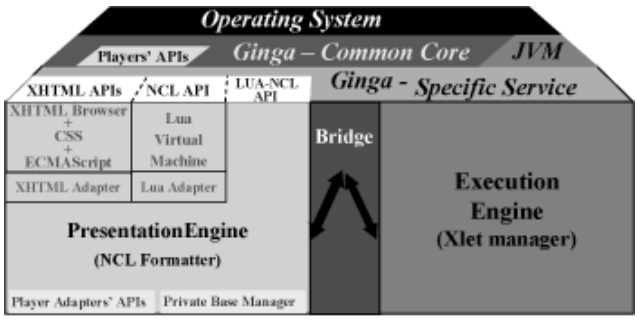


Figura 1.29. Arquitectura del middleware Ginga [8]

Como se puede observar, el primer módulo de referencia de Ginga es su núcleo común o Ginga-CC (Common Core), el cual se encarga de la decodificación de contenido común para las aplicaciones. Ginga-CC se encarga de la decodificación de contenido como JPEG, y MPEG, el acceso condicional a las aplicaciones, acceso al canal de retorno y la interfaz de usuario.

El segundo módulo de referencia de Ginga consiste en el motor de presentación, o Ginga-NCL, el cual es un sistema lógico para la ejecución de contenidos declarativos. El tercer módulo de Ginga-J procesa objetos Xlet. El Motor de ejecución Xlet permite pausar y reanudar una aplicación Java sin matar el proceso dentro de la máquina virtual.

Figura 1.30. Uso de tecnologías en los diferentes estándares de Televisión Digital.[3]

Capítulo II:

INTRODUCCIÓN A GINGA-NCL LUA

2.1 Librerías y paquetes del lenguaje descriptivo NCL Lua

2.1.1 Introducción

Como se mencionó en el capítulo anterior, el middleware que se utiliza en el estándar de televisión digital brasileño es Ginga, el cual permite ejecutar archivos de contenido desarrollados utilizando el lenguaje NCL (Nested Context Language), y a su vez, estructuras y algoritmos desarrollados en el lenguaje de escripting LUA.

Ginga es software libre, es decir de código abierto, que consiste en una capa de software que permite el desarrollo de aplicaciones para televisión digital terrestre, independientemente del hardware (posee un kernel Linux).

2.1.2 El lenguaje NCL

La historia del lenguaje NCL, y de Ginga tiene inicio en 1991. Ese año se desarrollaron algoritmos para solucionar problemas en sistemas Hipermedia utilizando el modelo de datos NCM (Nested context Model), sobre todo utilizado como una solución de la televisión en Alta Definición, pero al no ser aceptado (lo cual si ocurrió con MHEG, el cual se convirtió en el middleware para la televisión digital en Europa), fue el gobierno brasileño en 2003, que luego de las duras críticas sobre la intervención de Brasil dentro de los avances tecnológicos, quienes impulsaron su desarrollo, lo cual culminó en 2009 con la aceptación de Ginga-NCL como un estándar ITU.

NCL es un lenguaje desarrollado utilizando una estructura modular basada en web, por lo que permite una alta escalabilidad en aplicaciones. Es a su vez, un lenguaje declarativo, por lo que se especifica el funcionamiento multimedia, permitiendo la interactividad entre objetos de audio, video, y aplicaciones. La programación declarativa se basa en “declarar” condiciones, restricciones, y comportamientos que describen el problema, con lo que se puede intuir su solución, la cual es obtenida utilizando mecanismos de control, lo cual sucede en lenguajes como VHDL y Verylog.

Figura 2.1. Logotipo del lenguaje NCL [17]

El lenguaje NCL para aplicaciones de televisión interactiva basa su funcionamiento en la construcción de documentos hipermedia, dichos elementos se componen de nodos y enlaces para la sincronización entre archivos multimedia. Un nodo de contenido es un elemento multimedia (audio, video, imagen o texto). Al utilizar NCM como un modelo de contexto, permite anidar los elementos multimedia en una aplicación utilizando nodos de composición o de contexto.

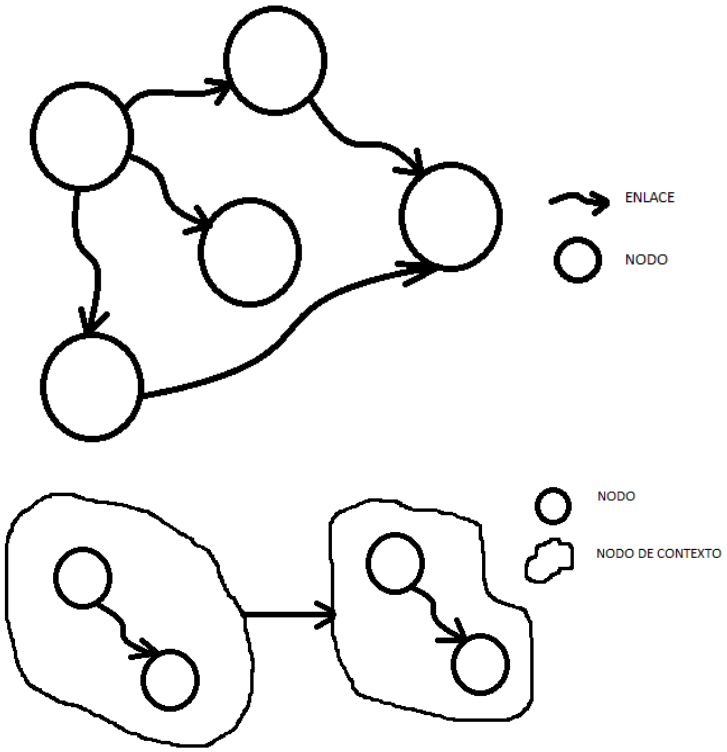


Figura 2.2. Descripción de un documento hipermedia utilizando nodos de contenido, nodos de contexto y enlaces [16]

Así, con NCM se representa una aplicación como una estructura de datos, y con los respectivos eventos asociados a cada elemento. Con NCL se logra un contexto

estructural en temporal, logrando especificar cuándo, cómo y qué se quiere mostrar en pantalla.

2.1.2.1 Diseño de Hipermedia

Para estructurar adecuadamente un documento hipermedia se deben separar las instancias de los elementos, es decir, separar sus características espaciales y temporales dentro del contexto de la aplicación. Así debemos especificar las siguientes características:

- Definir la categoría del elemento que se quiere reproducir, ya sea un elemento de audio, video, una imagen o un diálogo de texto
- Definir su característica espacial, es decir, la región de la pantalla en la que se va a desplegar dicho elemento
- Definir su característica procedimental, es decir, cómo va a aparecer dicho elemento (con sombreado, bordes, tipo de reproductor, etc)
- Definir su característica temporal, es decir, en qué momento se quiere reproducir un elemento (antes o después de una secuencia o de presionar un botón en el STB)

2.1.2.1.1 Categorización

En una aplicación multimedia debemos escoger el tipo de archivo o de elemento que se quiere reproducir. Este elemento, como se mencionó anteriormente, toma el nombre de nodo de contenido o nodo multimedia. Cada uno de estos nodos se define dentro de un contexto, para el caso de NCL se utiliza el cuerpo o body, es decir, el cuerpo engloba a un conjunto de nodos multimedia.

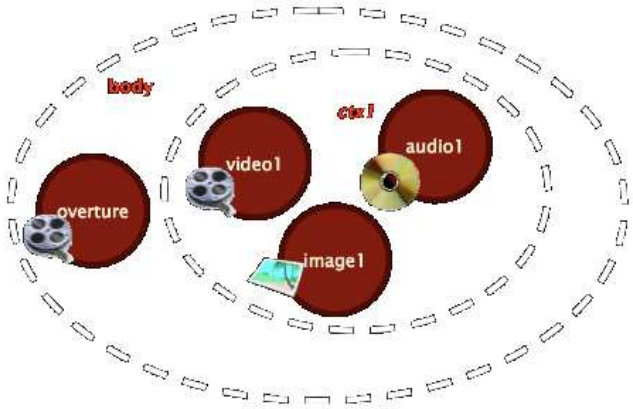


Figura 2.3. Nodos, contexto y body de un archivo Hipermedia [15]

2.1.2.1.2 Característica espacial

Una vez elegido el tipo de archivo a reproducir, se prosigue con su característica espacial, es decir, dónde se quiere que el elemento aparezca en pantalla. Para esto debemos definir el área en donde cada elemento multimedia se reproduce, utilizando para esto una región de reproducción.

Las regiones indican la posición y la dimensión de un elemento multimedia, la cual puede especificarse en porcentajes o en número de pixeles. Normalmente debido al formato de videos en HD y SD se utilizan regiones rectangulares. Dicha región define el área donde se pueden visualizar los elementos multimedia, y para realizar esta asociación objeto-región se utiliza un descriptor.

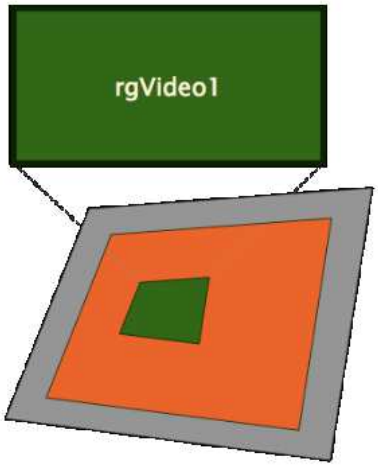


Figura 2.4. Región multimedia [15]

2.1.2.1.3 Característica procedimental

Una vez asociado el elemento multimedia a una región, se debe especificar cómo reproducir dicho elemento, y para esto se utilizan los descriptores. Un descriptor además se utiliza para definir el volumen, borde, y demás atributos de un elemento a ser reproducido. En la figura 2.5 se observa una representación de un descriptor, el está asociado a una región que será controlada.

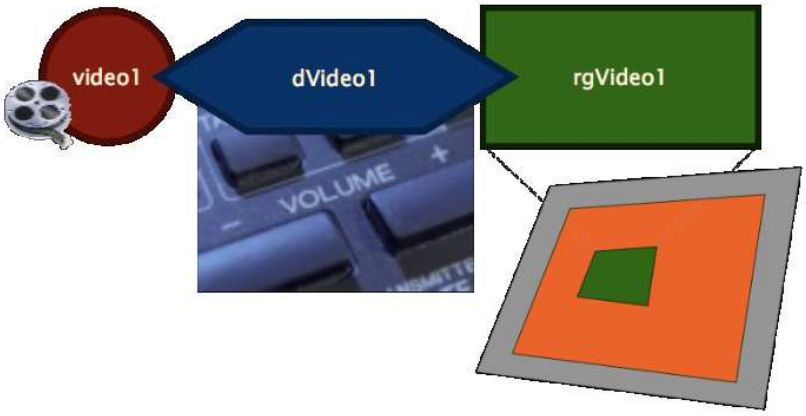


Figura 2.5. Descriptor asociado a su región [15]

2.1.2.1.4 Característica temporal

Finalmente, una vez conseguido el diseño de qué, y cómo reproducir, se debe agregar una característica temporal al elemento multimedia, es decir, cuándo se reproducirá en la aplicación. La analogía analizada es una puerta de reproducción, que permite que el nodo de contexto pueda reproducir su contenido.

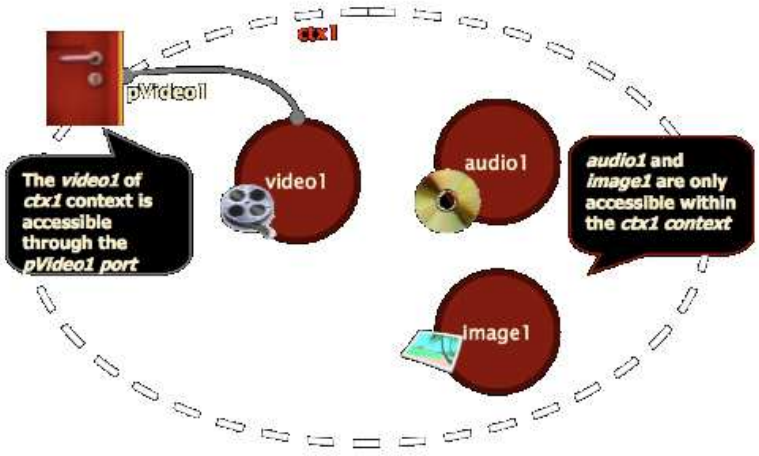


Figura 2.6. Acceso a un contexto mediante port (compuerta) [15]

2.1.2.2 Diseño NCL

La estructura básica de un documento NCL consta de los siguientes elementos:

1. Encabezado.- descripción XML del documento en general, donde consta la versión del compilador y el codificado para la programación. La primera parte (primera y segunda línea) describen el documento NCL, y la segunda parte hace referencia al encabezado del programa donde se definen descriptores, conectores y regiones (muy parecido a un archivo de encabezado en lenguaje C).
2. Cuerpo del programa.- es la sección denominada "body" donde deben definirse los contextos, los nodos y los enlaces entre los mismos.
3. Declaración de Regiones.- Donde se inicializan las regiones, sus dimensiones y las puertas de acceso a las mismas.

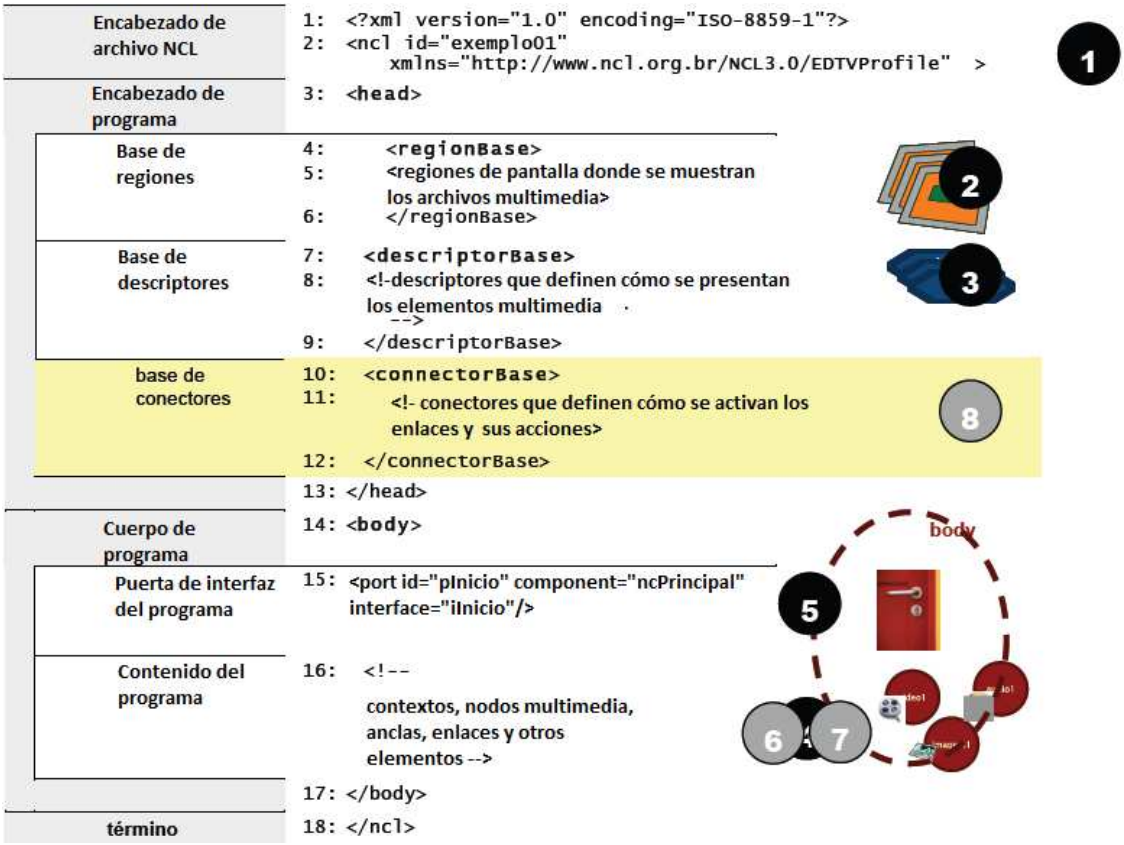


Figura 2.7. Estructura básica de un documento NCL [16]

En la figura 2.7 se puede observar un documento NCL básico y estructurado. Según [16] basándonos en este diagrama, para construir un documento NCL apropiadamente, se deben definir los siguientes elementos:

1. Los encabezados del archivo NCL y del programa
2. Definir las regiones de pantalla donde se presentarán los elementos multimedia (región Base)

3. Definir mediante descriptores cómo y dónde se presentarán los nodos multimedia (descriptor Base)
4. La asociación a los descriptores de los nodos multimedia (contenido) y la estructura (contextos) del documento.
5. Definir el inicio de la aplicación, es decir la puerta de interfaz o de entrada al programa, la cual debe apuntar al primer nodo que se quiere reproducir.
6. Definir un ancla para los nodos multimedia. Un ancla define un área y un atributo mediante el uso de nodos multimedia y sus enlaces.
7. Definir enlaces entre los nodos multimedia y nodos de contexto
8. Definir los conectores que especifican el comportamiento de los enlaces.

2.1.3 LUA

Lua es un lenguaje de scripting, desarrollado diseñado e implementado por la Pontificia Universidad Católica de Río de Janeiro en Brasil (PUC-Rio), el cual permite un tipo de programación imperativa, es decir, se basa en términos del estado de programa y sentencias para modificarlo utilizando instrucciones para llegar a una solución. Lua es uno de los lenguajes más utilizados en el mundo, pues es liviano y permite al programador realizar operaciones complejas. Lua es utilizado en programas profesionales como Adobe Photoshop, y juegos como World of Warcraft.

Figura 2.8. Logotipo del lenguaje Lua [17]

Lua es, en muchos aspectos, el lenguaje más poderoso, rápido y liviano de scripting, pues combina la sintaxis de scripting, con arreglos de semántica extendida basados en descripciones asociativas de datos. Una característica importante de Lua es que se ejecuta dentro de una máquina virtual, y maneja adecuadamente las sobrecargas de memoria, lo cual permite al programador migrar libremente entre plataformas, y despreocuparse de la administración de memoria en desuso.

Otra ventaja de Lua, es que además de ser la plataforma más rápida de scripting, es una de las más pequeñas, pues su código fuente es de apenas 860Kb, lo cual permite utilizar este lenguaje en procesadores embébidos, y aplicaciones electrónicas con memoria limitada.

Como se mencionó en puntos anteriores, Lua es un lenguaje de programación imperativo de scripting, es decir, que se basa en una secuencia de comandos para realizar operaciones o acciones sobre una sintaxis determinada. Dichos scripts son compilados a código bytecode y ejecutados en la máquina virtual de Lua basada en C y en Linux.

Al ser scripting, no tiene una estructura rígida a seguir pues depende de las características del problema que se quiera solucionar, pero sigue una sintaxis básica en cuanto a operadores, bucles y sentencias.

En parámetros generales, las convecciones léxicas dentro del lenguaje Lua pueden ser caracteres (strings) y utilizado como identificadores. Existen palabras clave reservadas y no pueden utilizarse como nombres de variables, como se muestra en la figura:

```
and       break     do        else      elseif
end       false     for       function  if
in        local     nil       not       or
repeat    return    then      true      until     while
```

Figura 2.9. Palabras restringidas en Lua [17]

Los valores y tipos en Lua tienen una característica especial, pues Lua es un lenguaje dinámicamente tipado, lo que significa que no existen tipos de variables, sino sólo tipos de valores. Los tipos de valores existentes son los siguientes:

- Nil.- ausencia de valor
- Boolean.- valor booleano true o false
- Number.- representación de números reales con coma flotante de doble precisión
- String.- caracteres
- Function.- funciones escritas en Lua o en C
- Userdata.- interfaz entre datos arbitrarios en C y en Lua. Uso de memoria y asignación de identidades.
- Thread.- representan procesos internos de Lua, y se utilizan para la implementación de co-rutinas.
- Table.- arreglos de cualquier tipo de valores.

La declaración de variables en Lua es muy simple, por ejemplo si se necesita inicializar una variable con el valor de nombre se lo realiza de la siguiente manera:

Var::=nombre

Las sentencias se denominan utilizando una unidad de ejecución llamada “Chunk”. Un chunk es un conjunto de sentencias que se ejecutarán, y se los representa de la siguiente manera:

Chunk::={sentencia[‘;’]}

Una vez inicializado, el chunk puede ser almacenado en un fichero para su posterior precompilación o compilarlo en forma binaria.

Las clásicas estructuras de control como if, while y repeat tienen una sintaxis similar a la programación en C. Por ejemplo para realizar un bucle while se lo hace de la siguiente manera:

Sentencia::= **while** exp **do** bloque **end**

De igual manera para un bucle if

Sentencia::= **if** exp **then** bloque {**endif** exp **then** bloque} [**else** bloque] **end**

Para una sentencia **for** el código a utilizar sigue la siguiente estructura:

sentencia ::= **for** nombre '=' exp1 ',' exp2 [',' exp3] **do** bloque **end**

2.1.4 Librerías y paquetes

2.1.4.1 Elementos del lenguaje NCL

NCL permite un acceso por módulos a su aplicación. Los módulos son conjuntos coherentes cuya combinación es un perfil del lenguaje. Existen dos perfiles definidos en el estándar brasileño:

- EDTV o Enhanced Digital TV Profile.- Uso de scripting Lua para correr aplicaciones interactivas
- BDTV o Basic Digital TV Profile.- Transmisión de televisión sin interactividad.

Como se mencionó anteriormente cada elemento o módulo del lenguaje NCL se compone de dos componentes básicas: el encabezado, el cuerpo del documento.

El encabezado o <head> es un directorio padre para elementos como

- <importedDocumentBase>
- <ruleBase>
- <transitionBase>
- <regionBase>
- <DescriptorBase>
- <meta>
- <metadata>

Donde:

<importedDocumentBase>.- especifica la utilización de un Documento NCL diferente (anidar documentos)

<ruleBase>.- es parte de las reglas de prueba, y designa las reglas de reproducción

<transitionBase>.-es un módulo de transición utilizado para el cambio de un estado de reproducción al siguiente.

<regionBase>.- define un conjunto de regiones <region>

<DescriptorBase>.- especifica la información para ejecutar cada elemento presente en el documento, designando atributos espaciales y temporales.

<meta> y <metadata>.- Utilizados dentro del módulo SMIL(Synchronized Multimedia Integration Language) de NCL. Permite manejar información sobre los elementos multimedia utilizados.

El cuerpo del programa o body, que se define como un nodo de contexto que puede tener enlaces (modelo NCM) puede contener elementos como:

- <port>
- <attribute>
- <media>
- <context>
- <switch>
- <link>

Donde:

<Port>.- especifica una compuerta o puerta de acceso a un elemento multimedia. Es un nodo que se mapea mediante una interfaz a componentes hijo.

<attribute>.- define un nodo de atributo o un grupo de nodos dentro de un contexto.

<media>.- define un objeto multimedia, integrando atributos como su tipo (audio, video, imagen o texto) y su localización (directorio en el programa).

<context>.- define los nodos de contexto. De igual manera un contexto puede tener atributos y elementos hijo según se requiera.

<switch>.- define nodos de documentos alternativos, para lo cual se debe basar su comportamiento en reglas <rule>.

<link>.- conectores entre los nodos de hipermedia y los roles que cumple dicho conector definido mediante <causalConnector>

2.1.4.2 Elementos de Lua

Dentro de NCL, Lua es un elemento multimedia de tipo "application/x-ginga-NCLua", que se ejecuta durante o luego de una acción realizada y registrada en un estado del elemento durante la aplicación. Gracias a la librería nclCommand de la clase de objeto NCLua, las funciones descritas en el documento Lua a ejecutarse, pueden estar referenciadas a un tiempo específico durante la reproducción, por lo cual permite que el desarrollador especifique el momento en el cual el comando sea ejecutado.

La librería nclCommand es un parámetro de tiempo de referencia, como toda función en C tiene sus atributos, el cual tiene un significado sobre el tiempo cuando se va a ejecutar una aplicación Lua o el tiempo que durará su ejecución. El parámetro en nclCommand ser entregado de tres maneras:

- Como un elemento vacío, lo cual indica a NCL que el comando Lua se debe ejecutar inmediatamente
- Como un valor numérico, con lo que NCL interpreta este valor como una cantidad de tiempo en segundos de retardo para que el comando Lua se ejecute.
- Como una tabla expresada con dígitos, indicando la fecha en la que se quiere que se ejecute el comando Lua. El formato de este aspecto se lo puede observar en la tabla 2.1

Tabla 2.1. Formato de ingreso de datos para nclCommand

Año	Mes	Día	Hora	Minuto	Segundo	ISDST(daylight saving flag)
4 dígitos	1-12	1-31	0-23	0-59	0-61	Boolean (true o false)

Existe también la librería "nclExtended", igual perteneciente a la clase NCLua. Esta librería permite definir acciones de ejecución, pausa, paro o resumen de funciones e

interfaces para objetos multimedia, por lo que el resultado generado de esta librería es una condición que se puede utilizar en NCL.

2.2 El compilador GINGA NCL

2.2.1 Comandos de edición

El middleware Ginga, como se observó se encarga de ejecutar el código NCL y mostrarlo en pantalla, así también como la decodificación de los elementos de audio y video. Para hacer esto el núcleo común de Ginga, que se muestra en la figura 2.10, se encarga de realizar las operaciones requeridas.

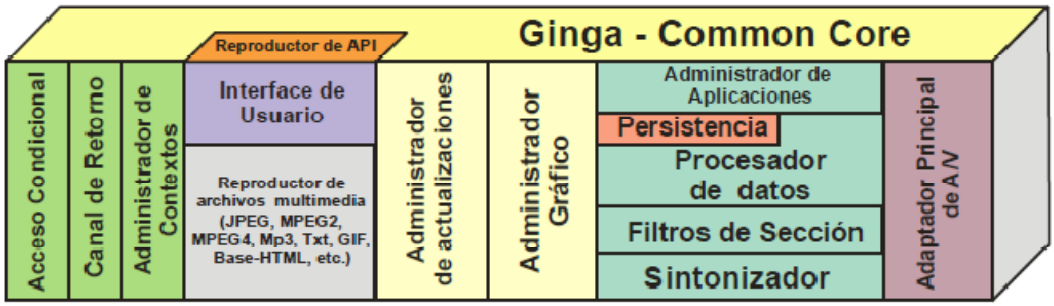


Figura 2.10. Núcleo Común de Ginga [15]

Como se puede observar, el núcleo se encarga principalmente de la reproducción de los archivos multimedia, y de las aplicaciones mediante el filtro de sección y el administrador de aplicaciones. Los componentes asociados únicamente al administrador de presentación de Ginga-NCL se muestran en la figura 2.11.

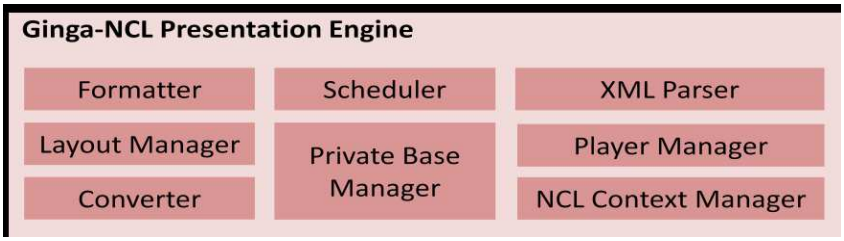


Figura 2.11. Motor de presentación de Ginga-NCL [15]

Los módulos del motor de presentación cumplen las siguientes funciones:

- Módulo Formateador.- Realiza la recepción de las aplicaciones NCL que vienen del Ginga-CC y su posterior control de ejecución.
- Módulo Scheduler (programador).- se encarga de la evaluación de todos los elementos que llegan del carrousel para organizar la reproducción del documento NCL. Organiza los elementos multimedia a ser ejecutados.
- Módulo XML Parser.- convierte la aplicación NCL a un lenguaje de datos para Ginga.
- Módulo Layout Manager (administrador de diseño).- mapea las regiones definidas en el código NCL

Ginga se vale de su intérprete NCL que reside en el procesador de datos, y el administrador de base privada. El intérprete NCL se encarga de controlar la presentación de todos los nodos en el tiempo especificado, y el administrador de base privada recibe los comandos de edición de dichos archivos NCL. Los comandos de edición relacionados se encuentran especificados en la norma ABNT NBR 15606-6, donde se establece eventos de flujo de NCL, con tiempos de vida de los archivos, lo que permite la edición de una aplicación NCL mientras se realiza la transmisión de audio y video.

Los comandos de edición ejecutados por el administrador de base privada se dividen en tres grupos:

1. Comandos de activación.- se encargan de activar y desactivar una aplicación NCL.
2. Comandos de reproducción.- grupo de comandos que controlan el flujo de una aplicación en curso, es decir, realizan el control de pausa, resumen, inicio y fin de la misma.
3. Comandos de actualización.- se encargan de actualizar las aplicaciones en tiempo real, lo que permite cambiar una presentación mientras un programa está al aire.

Estos comandos viajan a través del carrousel de objetos DSM-CC, lo cual permite la transmisión de los mismos de una manera cíclica. Para esto se debe transmitir el documento completo.

2.2.2 Instalación de la máquina virtual de Ginga

Existen dos métodos para instalar Ginga en una máquina virtual. El primero y más utilizado es mediante la descarga de la imagen en VmWare que provee Telemidia para el trabajo y desarrollo de Ginga alrededor del mundo, y el segundo es mediante la instalación de los paquetes y dependencias de los repositorios de Linux.

A continuación se dará una guía paso a paso de ambos tipos de instalación. Se debe tomar en cuenta que la máquina virtual utiliza el kernel de Linux por lo que se debe setear la dirección IP como en cualquier kernel de código libre.

2.2.2.1 Instalación de la máquina virtual en VmWare

La instalación de imagen de la máquina virtual de Ginga es el procedimiento más sencillo para iniciarse con la programación en NCL Lua. Además de la máquina virtual se requieren asistentes para cargar los archivos NCL a la misma. Los pasos a seguir para la instalación son los siguientes.

Paso 1

Se debe descargar la máquina virtual que ofrece la PUC-Rio de la siguiente dirección: www.ncl.org.br/ferramentas/ubuntu-server10.10-ginga-i386.zip y descomprimirla en cualquier lugar del disco duro. De preferencia se debe manejar un directorio común para todas las máquinas virtuales, lo cual facilita su administración.

Paso 2

Se ejecuta VmWare donde tendremos una interfaz como muestra la figura 2.12

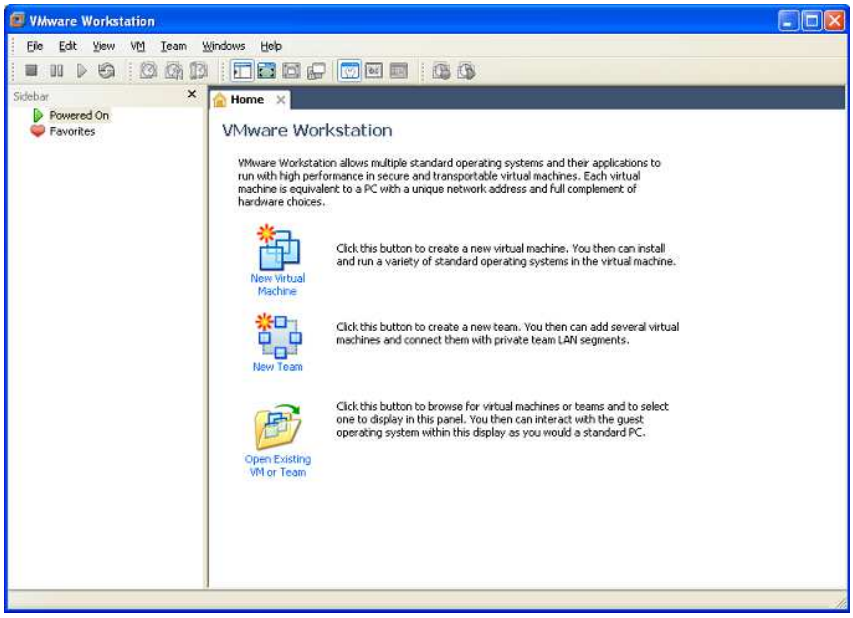


Figura 2.12. Pantalla principal de VmWare Workstation

Hacemos click en el menú “Open Existing VM or Team”, y en el menú que se despliega seleccionamos el archivo “ubuntu-server10.10-ginga-i386.vmx” que descomprimimos.

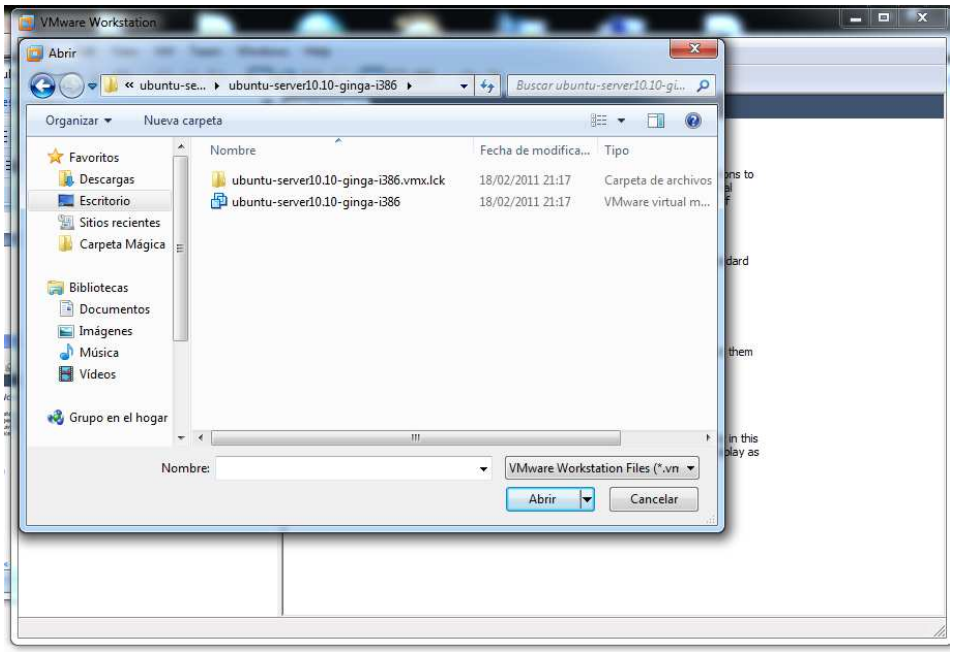


Figura 2.13. Selección de la máquina virtual de Ginga

Los atributos de la máquina virtual como memoria, disco duro e interfaces se seleccionan por defecto. Luego se inicializa la máquina virtual creada con el archivo ubuntu-server10.10-ginga-i386.vmx. La máquina virtual inicializará el kernel de Linux y mostrará luego de unos segundos estará lista para utilizarse.

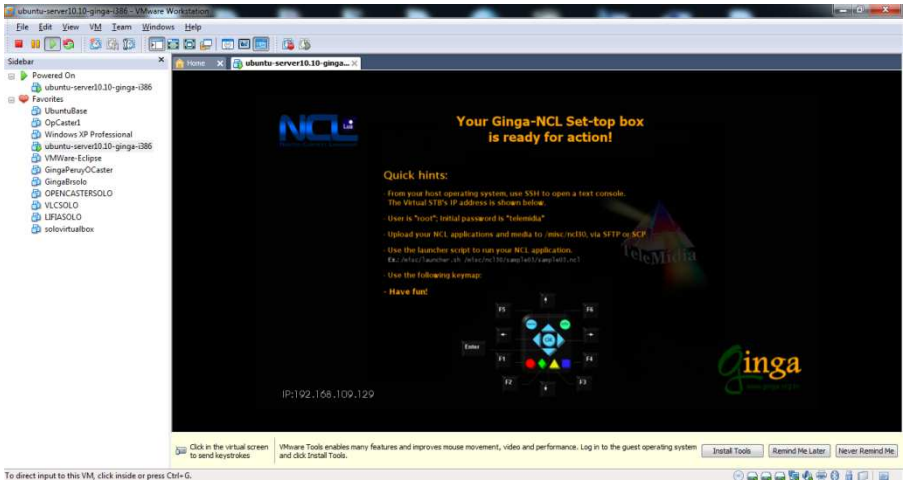


Figura 2.14. Pantalla principal de la máquina virtual de Ginga una vez ejecutada.

2.2.2.2 Instalación de la máquina virtual utilizando los repositorios de Linux

Para instalar Ginga desde los repositorios debemos tener en cuenta las dependencias que esto requiere. Los paquetes necesarios son los siguientes:

Librería de audio:

- libtool (Generic Library Support Script) (versión mayor o igual a 1.3.4)
- autoconf (Generador automático de guiones de configuración) (versión mayor o igual a 2.13)
- automake (Herramienta para crear archivos Makefile que cumpla con archivos GNU)(versión mayor o igual a 1.4)

Librería Telemidia:

Para compilar las librerías de Telemidia, debemos instalar los siguientes paquetes

- Ítem de lista desordenadaocaml-base-nox
- libfindlib-ocaml-dev
- libssl-dev
- camlp4
- libfindlib-ocaml
- libnewpki2

Librerías de Ginga:

- Ítem de lista desordenada Sistema make, multiplataforma y de código abierto
- Boost C++ Libraries development files
- Gtest – Google's framework for writing C++ tests
- luabind C++ for LUA
- uniones libcURL para LUA 5.1
- libcurl-ocaml-dev
- cmake-dbg

Entonces, debemos instalar las librerías requeridas de audio, para lo cual en Ubuntu se ejecutó el siguiente comando:

```
sudo apt-get install libtool autoconf automake g++ libssl-dev pkg-config libpng12-dev libdirectfb-dev cmake libavformat-dev libxine-dev liblua5.1-0-dev liblua5.1-socket2 libcurl4-openssl-dev libxerces-c-dev libboost-dev libdb4.7-dev libboost-
```

```
thread-dev libboost-filesystem-dev libboost-regex1.40-dev libavcodec-dev libxine1-ffmpeg libgtest-dev
```

Para poder reproducir los archivos de audio, es necesario instalar la librería FusionSound, con el siguiente comando

```
git clone git://git.directfb.org/git/directfb/core/FusionSound.git
```

Así deberíamos tener las Fuentes descargadas. El siguiente paso es compilarlas con el comando ./configure o con el comando ./autogen.sh. Con este comando compilamos la librería de sonido

```
cd FusionSound

./autogen.sh

make

sudo make install
```

Las fuentes de Ginga se las debe obtener del depositorio del Lifia, se debe descargar la subversión con el siguiente comando:

```
apt-get install subversion

svn checkout
svn://scm.ourproject.org/svnroot/ginga/middleware/ginga.ar/tags/release/1.1.0
```

La descarga de las librerías la realizamos con el siguiente comando

```
sudo ./autogen.sh --enable-graphics --with-directfb --enable-javascript --without-x --without-sd1 --prefix=/usr --enable-debuglevel=0

sudo make

sudo make install
```

La instalación de Ginga se la realiza de la siguiente manera:

```
cd 1.1.0/src/

sudo ./ginga-build.sh -i -S -P /usr/local -C /usr/local
```

y por ultimo configuramos la librería de visualización

```
sudo su

echo "/usr/lib/ginga" > /etc/ld.so.conf.d/ginga.conf

ldconfig
```

```
exit

echo "system=x11" > ~/.directfbrc
```

2.2.3 Interacción con la máquina virtual de Ginga

La máquina virtual de Ginga es un kernel de Linux modificado, y lo único que nos permite hacer es interactuar con ella mediante el teclado simulando un control remoto de televisión. Pero para cargar un archivo en la memoria es necesario realizar una conexión SFTP o SCP. . Para esto podemos hacer uso del terminal en Linux y abrir una conexión nueva en el menú “Lugares/Conectar a un servidor”, o si trabajamos en Windows deberíamos utilizar un cliente SCP adecuado para el caso. Como en el presente proyecto se trabajó en Windows, se explicarán los pasos para conectarse a la máquina virtual utilizando WinSCP.

Al abrir WinSCP se nos mostrará una ventana como la de la figura 2.15, en la cual debemos ingresar la dirección IP que se adjudicó a la máquina virtual por DHCP, el nombre de usuario, la contraseña de acceso y el puerto por el que se van a transferir los archivos.

Figura 2.15. Pantalla principal de WinSCP

La dirección IP la observamos en la pantalla principal de Ginga. El usuario y la contraseña de acceso son los siguientes:

Usuario	root
Contraseña	Telemidia

Una vez ingresados estos datos, se nos mostrará una ventana como en la figura 2.16 lo cual nos indicará que la conexión se realizó exitosamente.

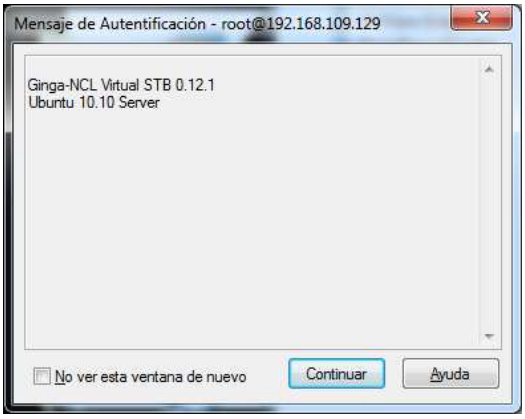


Figura 2.16. Autentificación de la conexión con la máquina virtual de Ginga

Hacemos click en Continuar, y se desplegará la pantalla mostrada en la figura 2.17. En el lado izquierdo del buscador tenemos los archivos del host, y en el izquierdo se tiene la estructura de archivos de Ginga.

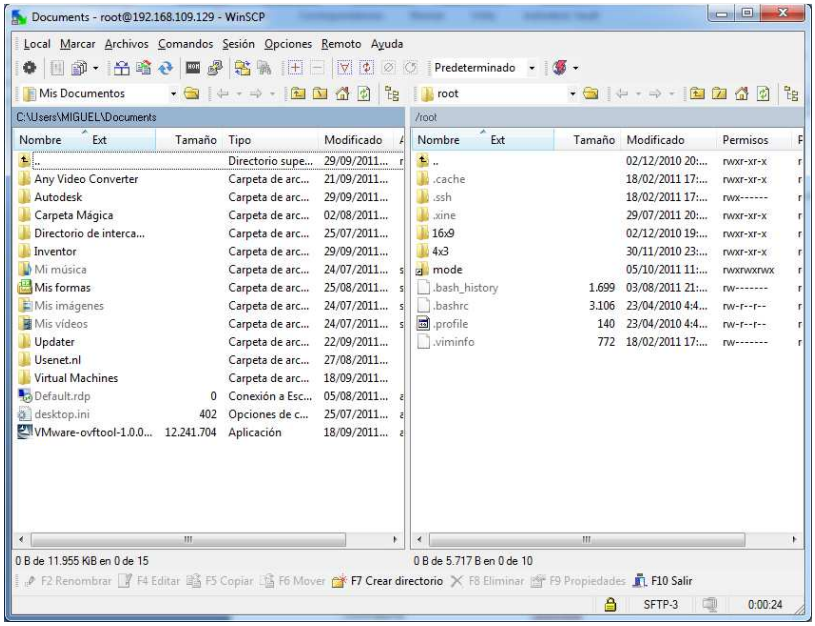


Figura 2.17. Estructura de archivos de Ginga

Ingresamos al directorio root/misc/ncl30 de la máquina virtual. En este directorio debemos guardar todas las aplicaciones NCL que queremos ejecutar, y como se verá posteriormente en esta carpeta también se guardará el archivo transmitido.

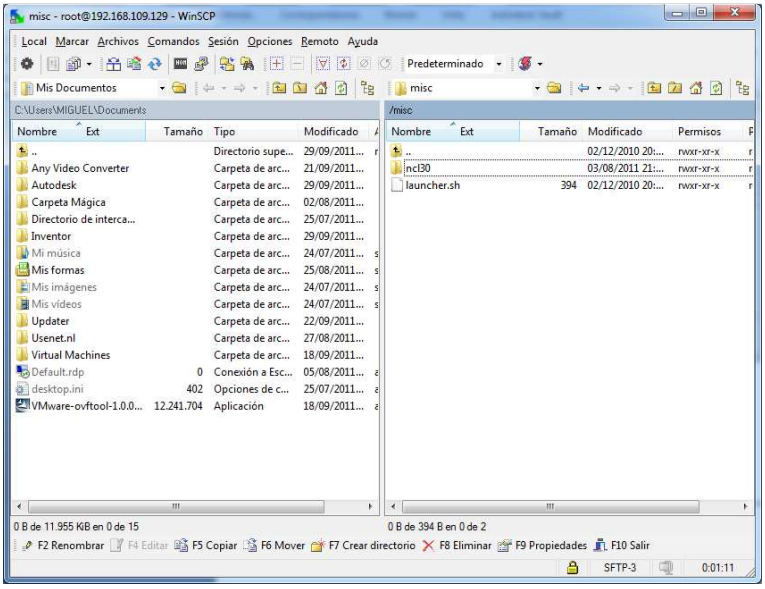


Figura 2.18. Directorio de ejecución de archivos NCL

Una vez que ingresemos al directorio ncl30, seleccionamos la aplicación NCL en el host y presionamos la tecla F5 para copiarla. Para ejecutar la aplicación NCL debemos hacer uso del protocolo SSH (Secure Shell) y de un cliente apropiado. SSH es un intérprete de órdenes para acceder a máquinas remotas. Este tipo de programas nos dan acceso a otras computadoras mediante una línea de comandos. En este proyecto se utilizó el cliente PuTTY de greenend.org.

La pantalla principal de PuTTY es la de la figura 2.19. Para ingresar a la máquina virtual debemos especificar la dirección IP de la misma.

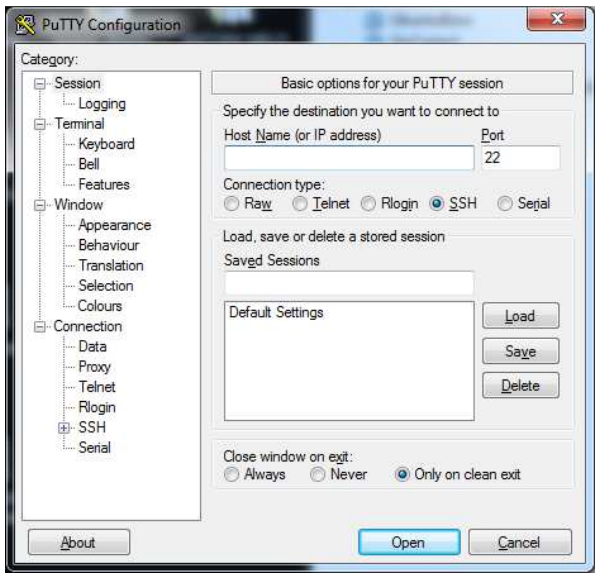


Figura 2.19. Pantalla de conexión de PuTTY

Una vez realizada la conexión, se despliega una ventana de comandos como la que se muestra en la figura. Para tener acceso total, se ingresa el usuario y la contraseña que se mencionó anteriormente.

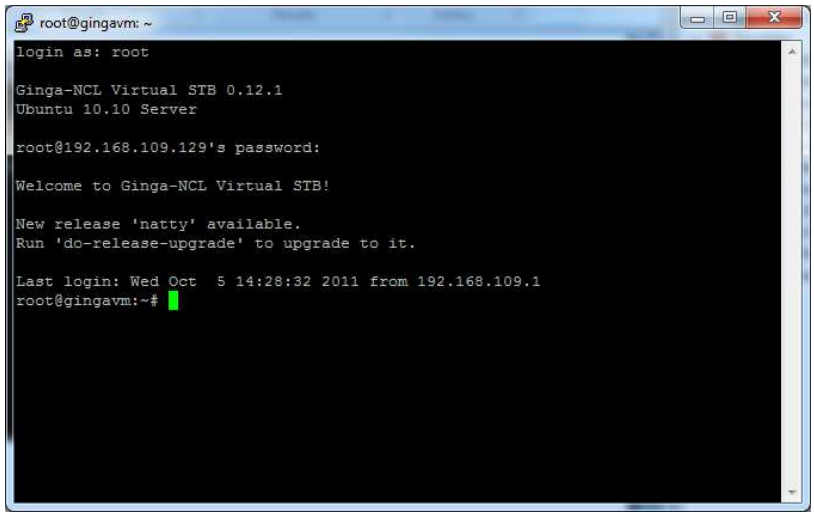


Figura 2.20. Ventana de comandos de la máquina virtual

Una vez dentro de la máquina virtual, debemos ingresar al directorio root/misc/ncl30 donde tenemos nuestra aplicación, y después ejecutarla con el siguiente comando:

Ginga –ncl miarchivoncl.ncl

Donde miarchivoncl.ncl es el nombre de la aplicación NCL que queremos ejecutar.

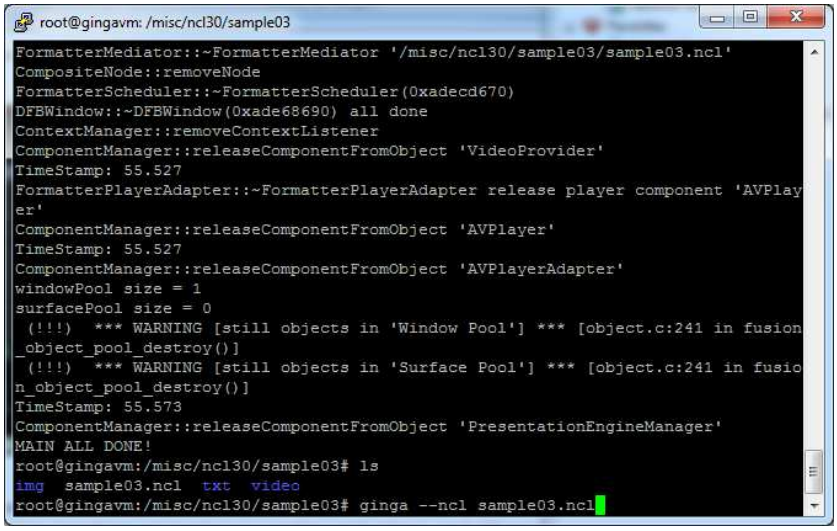


Figura 2.21. Ejecución de un archivo ncl

Si el archivo NCL no tiene problemas y si todas las librerías se ejecutaron correctamente, la aplicación se ejecutará como se muestra en la figura. La máquina virtual reproducirá los nodos multimedia indicados en el archivo NCL, y en la ventana de comandos se observarán dichas ejecuciones.

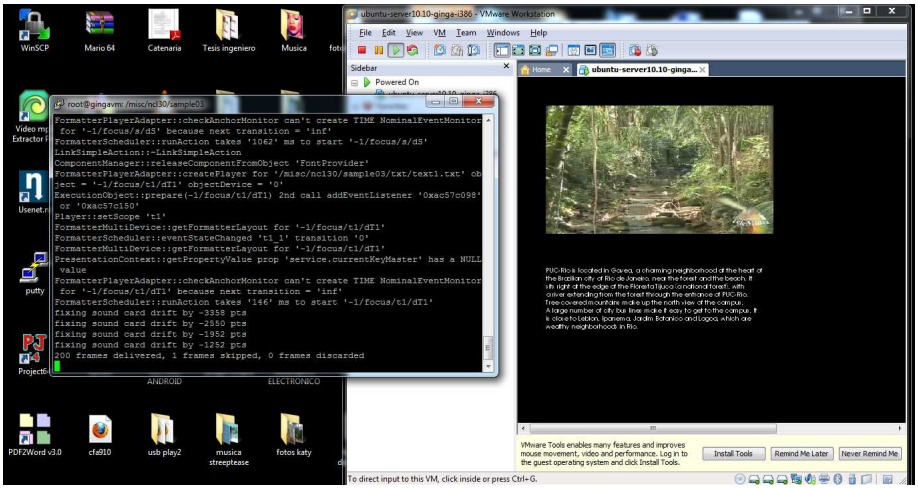


Figura 2.22. Aplicación NCL correctamente ejecutada.

2.3 Módulos y librerías que se utilizarán en el diseño

Lo primero que debemos considerar para cualquier diseño genérico en Ginga NCL, es la descripción de regiones y de los parámetros para los elementos multimedia. Estas características son generales para cualquier tipo de aplicación, y permiten al programador realizar un diseño estético y amigable para el usuario final.

Basándonos en la utilización de la interfaz gráfica de Java, NCL se maneja de una manera similar, pues así como en Java existe la clase GridBagLayout, su analogía en NCL es la región con su respectivo descriptor. NCL no posee módulos para la creación de interfaces, es decir, no posee una herramienta embebida que permita crear un menú, o un borde con un diseño por defecto, por lo que todas las imágenes, menús, e íconos para interacción se deben diseñar con un software apropiado, como por ejemplo Adobe Ilustrator, o alternativas de código abierto como Inkscape, Gimp y Blender.

(a)Inkscape (b)Gimp (c)Blender

Figura 2.23. Alternativas de Software libre para la edición de elementos multimedia

A continuación, se explicarán más a detalle los módulos y librerías que se utilizan en NCL para la creación de una aplicación genérica, y algunos módulos específicos para el canal de retorno y el manejo multimedia.

2.3.1 Regiones de reproducción multimedia

Como se mencionó anteriormente, una región consiste en el área de la pantalla en donde se reproducirá un elemento multimedia. Una región para que se ejecute en la pantalla debe estar definida en el encabezado del documento NCL, la cual puede ser la principal (que ocupa toda la pantalla y definida como una región base <regionBase>) o anidada a otras regiones secundarias (en el caso de requerir que más de un elemento multimedia se ejecute).

Una región en un documento NCL se define de manera básica mediante su identificador (id), el alto y el ancho que ocupará en la pantalla, como se muestra a continuación:

```
<región id=”regPrincipal” width=”1024” height=”480”>
        <región id=”regSecundaria” width=”480” height=”320”/>
</region>
```

Así definimos una región principal “regPrincipal” que ocupará 1024 pixeles de alto y 480 pixeles de ancho como la región total del dispositivo donde se reproducirá la aplicación, y también se define una región secundaria “regSecundaria” para reproducir un elemento multimedia específico.

Los atributos que se deben dar a una región son los siguientes: [16]

- Id.- identificador de la región (nombre de la misma).
- Title.- título de la ventana en la que se ejecuta la región.
- Left.- coordenada en pixeles del lado izquierdo de la región.
- Top.- coordenada en pixeles del borde superior de la región.
- Right.- coordenada en pixeles del borde derecho de la región.
- Bottom.- coordenada en pixeles del lado inferior de la región.
- Width.- ancho de la región.
- Height.- alto de la región.
- Device.- hace referencia al dispositivo de reproducción multimedia de la región
- zIndex.-indica el orden de presentación de regiones que se encuentran unas sobre otras.

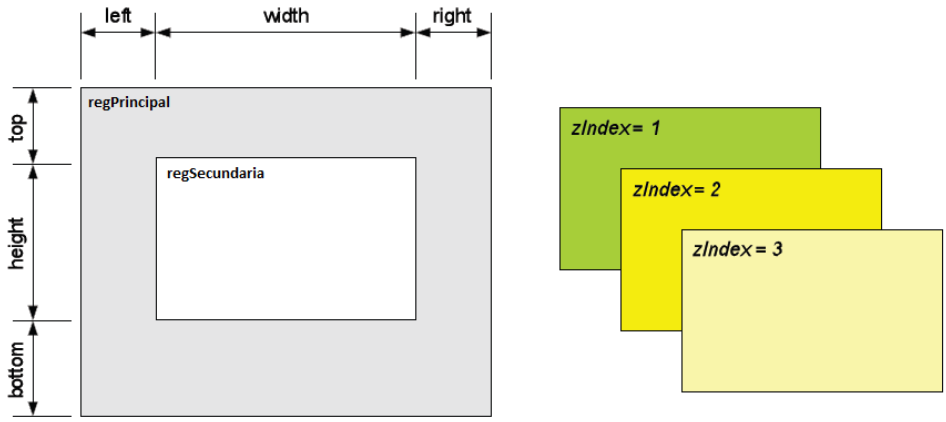


Figura 2.24. Atributos de una región [16]

2.3.2 Descriptores

Como se mencionó en ítems anteriores, un descriptor define la manera de presentar un elemento o nodo multimedia. Se debe tener en cuenta que el descriptor necesariamente asocia un nodo multimedia con una región. Para definir un descriptor se deben ingresar los siguientes atributos: [16]

- id.- nombre del descriptor
- player.- describe el decodificador que se encargará de la reproducción y presentación de un elemento multimedia
- explicitDur.- define el tiempo de duración en segundos de un elemento multimedia
- región.- expresa la asociación de un descriptor con su respectiva región
- freeze.- identifica una acción subsecuente al término de reproducción de un elemento multimedia, la cual mediante un valor booleano permite congelar el último cuadro de imagen en el caso de tratarse de un video.
- focusIndex.- de igual manera que zindex para regiones, focusIndex define la aplicación que se ejecutará o que tenga “foco”.
- focusBorderColor.- define el color del borde de un rectángulo sobrepuesto a una región.
- focusBorderWidth.- define el grosor del rectángulo sobrepuesto a una región
- focusBorderTransparency.- define un porcentaje de transparencia del borde sobrepuesto a la región.
- focusSrc.- define el archivo multimedia que se va a reproducir.
- focusSelSrc.- define un archivo multimedia que se ejecutará luego de presionar un botón.
- selBorderColor.- define el color de borde de un archivo multimedia que se ejecutará luego de presionar un botón.

- moveLeft.- permite cambiar el elemento de foco (focusIndex) al inmediato superior

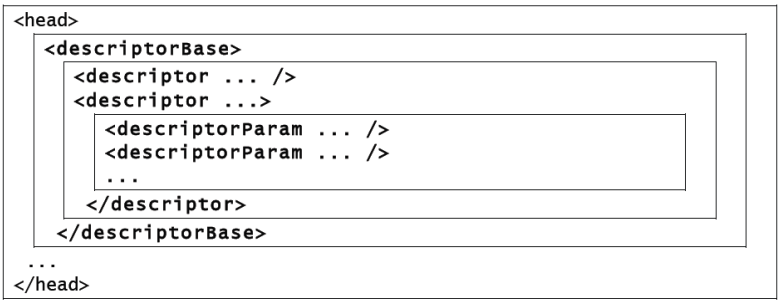


Figura 2.25. Esquema básico para la declaración de un descriptor [16]

2.3.3 Nodos multimedia

Un nodo de multimedia o nodo de contenido define el objeto (audio, video, imagen, texto, etc) que se utilizará dentro de una aplicación. Los atributos que se deben definir al crear un nodo multimedia son los siguientes:

- id.- Identificador o nombre del nodo multimedia
- Src.-directorio donde se encuentra el archivo multimedia a reproducirse, el cual puede ser un archivo local, un archivo remoto (http), un stream del canal de retorno(Rstp, o RTP), o un stream recibido de la capa de trasporte MPEG (lsdtv-ts)
- Descriptor.- identificador del descriptor anidado al nodo
- Refer.- referencia a otro nodo para su posterior reutilización
- type.- tipo de archivo dentro del nodo a reproducir. En la tabla se muestran los tipos de archivos soportados por Ginga-NCL.

Tabla 2.2. Tipos de archivos multimedia soportados.

Tipo		**Extensión**
Texto	html	.htm, .html
	css	.css
	xml	.xml
Imagen	bmp	.bmp
	png	.png
	gif	.gif
	jpeg	.jpg

Audio	basic	.wav
	mp3	.mp3
	mp2	.mp2
	mpeg4	.mp4, .mpg4
Video	mpeg	.mpeg, .mpg
Aplicación	application/x-ginga-NCLua	.lua
	application/x-ginga-NCLet	.xlt, .xlet, .class
	application/x-ginga-settings	-
	application/x-ginga-time	-

2.3.4 Contextos

Un contexto o también llamado nodo de composición, como se mencionó anteriormente, estructuran el documento hipermedia en NCL. Los atributos de un contexto son los siguientes:

- Id.- identificador o nombre
- Refer.- referencia a otro contexto

Un caso especial de contexto es el cuerpo "body" del documento NCL, y la manera de definir un contexto común es de la siguiente manera:

```
<body>
        <port id=”pVideoInicio” component= “videoInicio”/>
        …
</body>
```

2.3.5 Puertos

Un puerto o puerta de acceso, es un punto de acceso de un contexto a un contenido multimedia (contenido o nodo interno). Este punto de interfaz tiene los siguientes atributos:

- Id.- identificador o nombre
- Component.- nodo multimedia o contexto al cual hace referencia
- Interfaz.- nombre de la interfaz de destino

En la figura 2.26 se muestra la funcionalidad de un puerto de enlace.

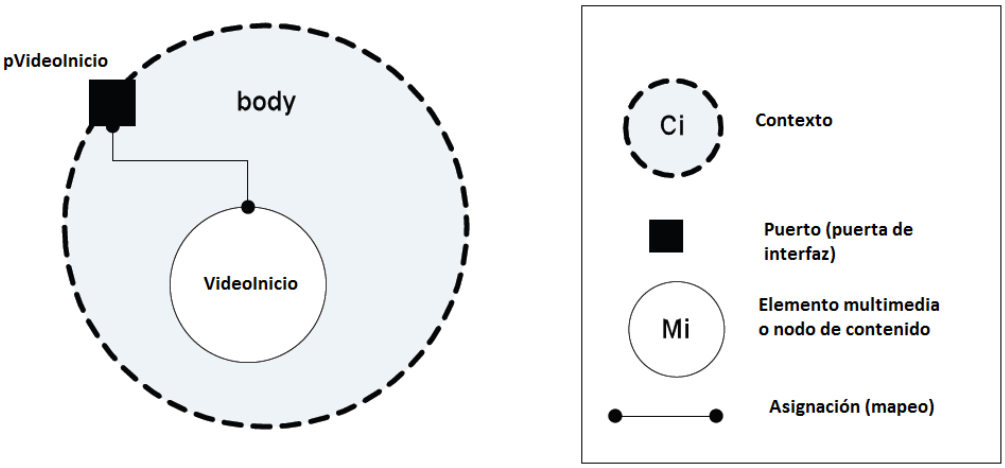


Figura 2.26. Funcionalidad de un puerto de enlace [16]

Un puerto (dentro de un contexto) se define de la siguiente manera:

```
<contexto>
        <port id= “pVideoInicio” component= “VideoInicio”/>
        <port id= “pImgInfo” component= “ImgInfo”/>
        …
<contexto/>
```

2.3.6 Conectores

Los conectores son los elementos encargados de definir causalidades entre los nodos de contexto. Todos los conectores se definen dentro de la base de conectores <connectorBase> y según [16] “Definen los papeles (roles) que los nodos de origen y destino ejercen en los enlaces que utiliza el conector”. En la figura 2.27 se muestra la funcionalidad de un conector, se puede hacer una analogía con una compuerta lógica, que utiliza causalidades de inicio y fin para describir el comportamiento de su salida.

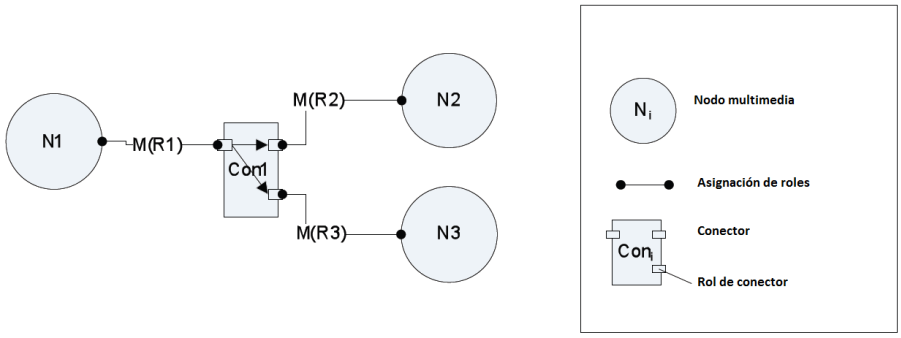


Figura 2.27. Funcionalidad de conectores [16]

La especificación de un conector causal <causalConnector>, define las condiciones para que un nodo se inicialice, es decir, mediante comparaciones permite agregar una condición causal a un nodo, y en su otro extremo define la acción a tomar cuando la condición se cumpla. Las condiciones y acciones que se pueden definir se muestran en la tabla 2.3:

Tabla 2.3. Condiciones y acciones definidas en un conector

Tipo	Rol	Descripción
Condición	onbegin	Inicio de la presentación del nodo multimedia
	onEnd	Final de la presentación del nodo multimedia
	onAbort	Final causado de la presentación del nodo multimedia
	onPause	Pasa de la presentación del nodo multimedia
	onResume	Ejecución de la presentación del modo multimedia en una pausa
	onSelection	Selección del nodo multimedia al presionar una tecla en el control
	onBeginAttribution	Inicio de la atribución de un valor al nodo multimedia
	onEndAtribution	Final de la atribución de un valor al nodo multimedia
Acción	start	Inicio de la presentación de un nodo multimedia asignado
	stop	Final de la presentación de un nodo multimedia asignado
	abort	Final abrupto de la presentación de un nodo multimedia asignado
	pause	Pausa de la presentación de un nodo multimedia asignado
	resume	Ejecución de la presentación de un nodo multimedia asignado
	set	Establecimiento de un valor para un nodo multimedia asignado

2.3.7 Enlaces

Los enlaces o links asocian a los nodos multimedia a través de un conector, es decir son las uniones entre el nodo y su respectivo conector. En NCL se describen dos tipos principales de enlaces: el enlace LinkParam para definir los parámetros del

conector, y el enlace Bind que indica el nodo multimedia al que hace referencia el enlace. Los atributos que se deben definir para un enlace son los siguientes:

- Id.- identificador o nombre del enlace
- Xconnector.- identificador del conector asociado a dicho enlace

Un enlace tiene una estructura básica como se muestra a continuación:

```
<link id= “nombreLink” xconnector= “idLink”>

    <bind role= “idRolCondicion” component= “idObjetoMultimedia” interface= “idInterfaz”/>

</link>
```

2.3.8 Anclas

Un ancla es un punto de entrada para un nodo multimedia. Define la utilización de segmentos o propiedades de un nodo para su posterior ejecución. Las anclas pueden ser de contenido, si definen un segmento de un elemento multimedia, o de propiedad si definen propiedades específicas del elemento multimedia como volumen, coordenadas de exhibición, dimensiones, etc. Los atributos de las anclas se definen a continuación:

- Id.- identificador o nombre del ancla
- Begin.- valor en segundos desde el cual se creará el ancla
- End.- valor en segundos donde terminará el ancla
- Dur.- valor en segundos que especifica la duración del ancla
- Coords.- valor en pixeles de un ancla
- Text.- texto del ancla en el archivo de origen
- Position.- posición del texto del ancla en el archivo de origen
- anchorLabel.- identificador del ancla en el archivo de origen

La estructura común de un elemento ancla se define a continuación:

```
<media type= “video” id= “videoBase” src= “directorio/videoBase.mpg” descriptor= “dVideo”>

        <!-anclas de contenido>

        <area id= “area1” begin= “5s” end= “9s”/>

        <area id= “area2” begin= “10s” dur= “22.3s”/>

        <!-anclas de propiedad>
```

```
        <property name= “top”/>

</media>
```

2.3.9 Switch

Un switch consiste en un contexto que posee varios nodos multimedia, de los cuales se puede elegir cuál se ejecutará dentro de la aplicación. Un switch ejecutará un nodo basándose en reglas de ejecución (al igual que su análogo en lenguajes comunes como Basic o C).

Antes de especificar un Switch, debemos definir las reglas y sus operadores. Una regla se define en la sección <ruleBase> y los operadores que la designan pueden ser los siguientes:

- eq.- igual
- ne.- no igual
- gt.- mayor que
- ge.- mayor o igual que
- lt.- menor que
- le.- menor o igual que

La estructura básica de una regla se muestra a continuación:

```
<ruleBase>

        <rule id= “rEn” var= “idioma” comparator=”eq” value=“en”/>

        …

</ruleBase>
```

Un switch, valiéndose de las bases especificadas se ejecutará cuando una de estas sea válida, y se describe de con la siguiente sintaxis:

```
<switch id=”switchAudio”>

        <bindRule rule=”rEn” constituent=”audioEn”/>

        <bindRule rule=”rSp” constituent=”audioSp”>

        <mediatype=”audio”          id=”audioEn”          src=”dir/audioEn.mp3”
descriptor=”dAudio1/”>

        <mediatype=”audio”          id=”audioSp”          src=”dir/audioSp.mp3”
descriptor=”dAudio2/”>

</switch>
```

Capítulo III:

DISEÑO DEL SOFTWARE PARA UN CANAL DE TELEVISIÓN DIGITAL

Introducción

El diseño de una aplicación para televisión interactiva, depende mucho de la finalidad de dicha aplicación, y de los requerimientos del televidente. No existe un modelo predeterminado para el diseño de una aplicación, ni tampoco una lista de requerimientos de un canal específico, pues las cadenas de televisión basan su programación en lo que creen les dará más utilidades.

Por lo tanto quizá la única regla que se debe manejar en el diseño de la aplicación, es que sea amigable con el usuario final, fácil de utilizar, entretenida y estéticamente aceptable. Así también debe tener requisitos de mantenimiento y escalabilidad, con lo que el proveedor de servicios será capaz de modificar la aplicación según el comportamiento del mercado.

Se basará el diseño de la aplicación para un canal de deportes en la opción de poder cambiar ciertos parámetros de la misma, como son textos, imágenes de los menús, imagen del botón de interactividad, y otras características básicas, pero un cambio radical en la configuración de la aplicación, requiere el conocimiento del lenguaje NCL, y un rediseño de la manera como se reproducen los contenidos.

En el presente capítulo, se especificarán las consideraciones que se tomaron para el diseño de la aplicación requerida, más no se intenta dar una guía de programación, ni una lista de requerimientos, pues los mismos son muy cambiantes, difieren regionalmente, y dependen de gran manera de las condiciones sociales y económicas de los usuarios finales.

3.1 Análisis de los requerimientos de un canal de deportes sin retorno

Para éste análisis, nos basaremos en las condiciones actuales de canales de televisión conocidos a nivel de Latinoamérica, como es FoxSports e ESPN. Se propondrá una solución para la interactividad del usuario, y la plataforma necesaria para realizarlo.

3.1.1 Red de transmisión de televisión digital

Un sistema de televisión en general tiene una estructura típica conocida como topología en árbol, con la cual se brinda el servicio de televisión, ya sea abierto o pagado. Un canal de televisión común tiene cinco partes principales [19]:

- Cabecera,
- Red troncal,
- Red de distribución,
- Acometida
- Equipos terminales (equipo de suscriptor).

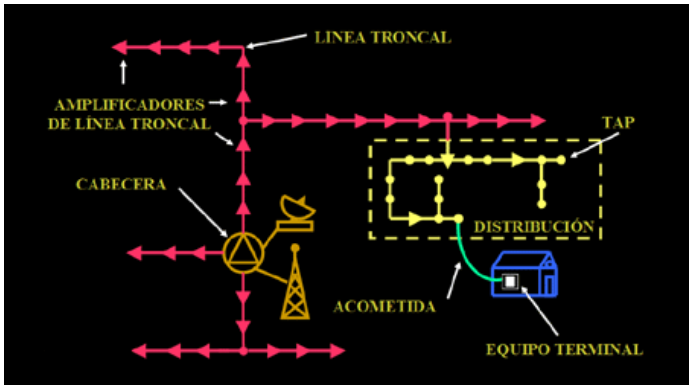


Figura 3.1. Red común de televisión.[19]

3.1.1.1 Cabecera

La cabecera es el punto de origen de la señal a transmitir, es decir es el conjunto que compete los equipos de edición, y los equipos de transmisión.

Los equipos de edición son computadores, que mediante software libre o software propietario, capturan las imágenes que se transmitirán y las editan agregándoles contenido audiovisual. Además, los equipos de edición, engloban la gestión web del canal de retorno, el Playout de audio y video, y la gestión de las aplicaciones interactivas.

Los equipos de transmisión son los equipos necesarios para convertir la señal de televisión en una señal de radiofrecuencia. Además abarca la multiplexación de los contenidos para enviarlos por el carrousel de objetos en Ginga.

Con esto, los requerimientos básicos para una cabecera de un sistema de televisión son las siguientes:

- Para video:
  - Cámara de video
  - Micrófono
  - Equipo de iluminación
  - Trípode
  - Set apropiado
- Para la Estación de edición
  - Computador
  - Monitores de alta definición
  - Switch o Router para trabajo en red (depende de la organización de la red)
- Para audio
  - Mesa de sonido
  - Micrófonos direccionales y pies para los mismos
- Almacenamiento
  - Discos duros

La mejor opción para implementar una cabecera de televisión digital, es la que se muestra en la figura 3.2.

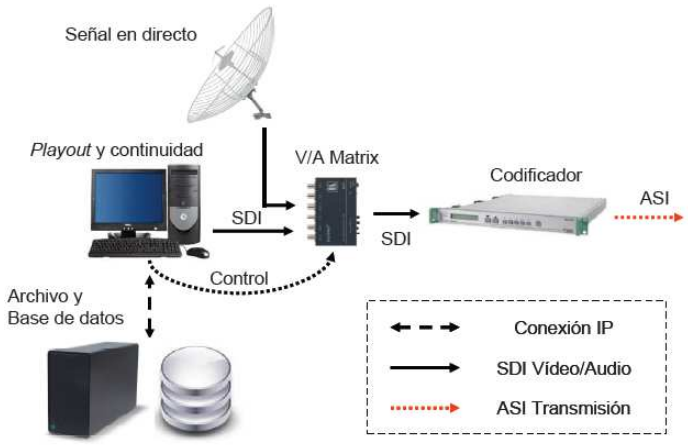


Figura 3.2. Solución de una cabecera para televisión digital [20]

3.1.1.2 Red troncal

La red troncal, es la red de transporte para la señal. Dependiendo de la estación emisora, esta puede ser por cable coaxial, Ethernet, fibra óptica o por

radiofrecuencia. En el caso de un canal abierto, la modulación COFDM es por radiofrecuencia, por lo cual la red troncal se compone de equipos amplificadores y repetidoras.

3.1.1.3 Red de distribución

La red de distribución es la interfaz entre el usuario y la red troncal. Mediante esta red se realiza el control de usuarios (por zonas) en televisión por cable. De igual manera, en televisión abierta normalmente no se utilizan cables en la red de distribución, pues la señal llega por radiofrecuencia al equipo del usuario final. Para canales pagados, esta señal llega al consumidor mediante la red del proveedor de servicios, la cual puede ser de cable coaxial o fibra óptica, exceptuando el caso de televisión satelital.

3.1.1.4 Acometida

La acometida, hace referencia al equipo con el cual el usuario capta la señal del transmisor. En televisión por cable consiste en un Tap (unión) con el cable de distribución, mientras que en televisión abierta y satelital es la antena.

3.1.1.5 Equipo terminal

El equipo terminal, es el hardware de acondicionamiento de la señal para que sea reproducida en un televisor. En sistemas de cable y satelital es un decodificador (ahora digital), mientras que en la televisión analógica antigua era el televisor como tal, ahora es el STB.

3.1.2 Diseño de una plataforma digital

Se analizarán dos soluciones para una plataforma de software para la transmisión para un canal de televisión digital terrestre. La primera se realizará utilizando software propietario y soluciones profesionales, y la segunda utilizará software abierto. La diferencia de costos entre ambas es muy significativa, por lo que la

elección de cualquiera de las dos opciones dependerá del presupuesto de la empresa que desee brindar este tipo de servicio, y el mercado al cual irá dirigido.

En la figura 3.3 se muestra el esquema básico de transmisión y recepción para un canal de televisión digital basándose en lo descrito en el tema anterior.

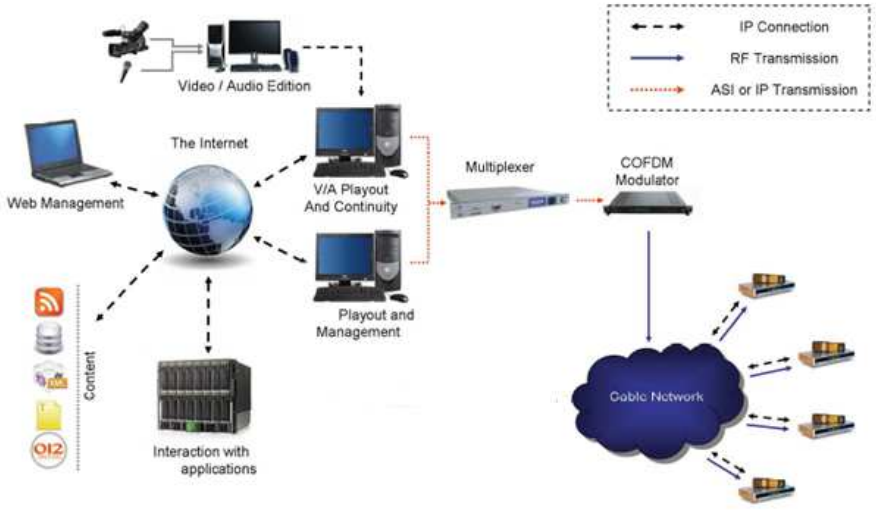


Figura 3.3. Esquema de transmisión de una red de televisión digital [20]

3.1.2.1 Solución con software propietario

Para diseñar una plataforma para la transmisión de televisión digital, debemos centrarnos en los programas de gestión. Para esto se divide la cabecera en las siguientes partes:

- Sistema de edición de audio y video
- Sistema de gestión, almacenamiento y playout
- Sistema multiplexor de contenido

Además se hará un pequeño análisis de los elementos básicos de hardware como el equipo de playout, el multiplexor, y el modulador, los cuales se incluirán en la respectiva cotización de la solución planteada.

3.1.2.1.1 Sistema de edición de audio y video

La solución más utilizada para la edición de archivos de audio y video es Adobe Premiere de Adobe.

Figura 3.4. Logotipo de Adobe Premiere [21]

Características [21]:

- Trabajo en tiempo real
- Producción eficiente con un flujo de trabajo basado en archivos nativos
- Compatibilidad con formatos mpeg4, mpeg2 HD
- Flujo de trabajo de metadatos integral
- Incrustador acelerado para GPU gráfica
- Edición no lineal en tiempo real
- Corrección avanzada de color
- Administrador de proyectos
- Procesador de video YUV
- Líneas de tiempo múltiples
- Mezclador de audio con soporte para sonido sorround 5.1
- Integración con Photoshop, Encore y After

3.1.2.1.2 Sistema de gestión, almacenamiento y playout

El sistema de playout y gestión es un paquete de software encargado de preparar y codificar las aplicaciones y contenidos para que éstos puedan ser transmitidos por la red [2]. Para elegir un sistema apropiado para gestión y playout de aplicaciones se tomarán en cuenta los siguientes aspectos:

- Capacidad de la plataforma para gestionar stream NCL, MHP, y bases de datos
- Interfaz gráfica
- Stream MPEG2 de la trama de transporte

Uno de los programas más utilizados en España, Brasil y Argentina [20], es VSNSharer de la empresa VSN (Video Stream Networks). Es un software propietario que consiste en un gestor de contenidos en línea dentro de una red de video. Entre sus características principales, permite a las estaciones de edición, equipos de postproducción y de contenido en vivo, encontrar un video o una aplicación dentro de un servidor.

Figura 3.5. Logotipo de VSN [22]

3.1.2.1.3 Sistema multiplexor de contenido

Un multiplexor para televisión digital es un instrumento que posee varias entradas de video, o soporta varios streamings de entrada, y tiene una salida DVB-ASI para su posterior conexión con el modulador. La función de un multiplexor, como se mencionó anteriormente, es multiplexar las tramas de transporte de video, y de aplicación, en una sola trama MPEG2. Para seleccionar adecuadamente una solución para un multiplexor se tomaron las siguientes consideraciones:

- Estabilidad del instrumento, pues requiere funcionar continuamente sin interrupción
- Interfaz de usuario fácil de manejar
- Interacción por comandos para formar la trama de transporte requerida

La solución hardware software que se ha seleccionado, pues cumple las características requeridas, es MuxXpert SDK de la empresa Dektec.

Figura 3.6. Logotipo de MuxXpert [23]

MuxXpert ofrece características como:

- Multiplexación en tiempo real
- Librerías .Net, C++ y VB.NET para trabajo en red
- Re-multiplexación de procesos SI
- Capacidad de introducir guías de programación (EPG)

- Control desde una máquina remota

La tarjeta de hardware utilizada con el sistema MuxXpert es la DTA-112 de la empresa Dektec. Esta tarjeta electrónica de modulación ofrece las siguientes características:

- Modulación QAM
- Rango de frecuencia de 47Mhz a 862Mhz
- Ancho de banda de 5 a 8MHz
- Relación señal a ruido de 44dB
- Entrada BNC para cable coaxial
- Salida QAM de -31.5 dBm programable

Figura 3.7. Tarjeta de modulación Dektec DTA-112 [23]

Para tener una idea más clara sobre el costo de una cabecera para televisión digital, en la tabla 3.1 se muestra una cotización de hardware y software requeridos para una solución utilizando software propietario, y equipos dedicados a la multiplexación.

Tabla 3.1. Costo de cabecera utilizando software propietario

Referencia	Item	Cantidad	Precio Unitario	Precio Total
Sistema de edición	Adobe Premiere	2	$ 285,00	$ 570,00
Sistema de playout	VSNSharer	1	$ 15.800,00	$ 15.800,00
sistema multiplexor	MuxXpert	1	$ 2.900,00	$ 2.900,00
Sistema modulador	DTA-112	1	$ 950,00	$ 950,00
Computadores	-	3	$ 1.150,00	$ 3.450,00
		Total		$ 23.670,00

3.1.2.2 Solución con software libre

Debido a las características del mercado actual, y las tendencias de software libre en el mundo de la informática, una solución de código abierto u OpenSource es necesaria para cualquier tipo de problema, y sobre todo cuando trabajamos con sistemas de televisión digital abiertos como es Ginga. A continuación se analizará una solución basada en software libre junto con los elementos de hardware para este propósito. Cabe recalcar que las tarjetas de modulación descritas en este subcapítulo utilizan software propietario, por lo cual una solución de Hardware libre es imperativa, pero el presente proyecto no abarca el diseño de la misma.

El objetivo es conseguir una emisión programada de los contenidos del canal de televisión, lo cual se logrará utilizando VLC y OpenCaster. El sistema funcionará como se muestra en la figura 3.8:

Figura 3.8. Esquema básico para la transmisión de un canal de televisión digital utilizando software libre

3.1.2.2.1 Sistema de edición de audio y video de código abierto

En un sistema de televisión digital la fase de edición se puede hacer con programas de código abierto como Lives o Cinilerra

(a) (b)

Figura 3.9. Programas de edición de video para Linux [24]

La plataforma sugerida de mayor soporte y prestaciones es Lives, la cual nos ofrece las siguientes características: [24]

- Generales:
  - Edición en vivo de audio y video
  - Permite la modificación de funcionalidades mediante pluggins que se pueden escribir en Perl, C, Python.
  - Control mediante OSC
- Video
  - Compatible con mpeg3 y mpeg4 HD
  - Streaming de video de entrada y salida
  - Soporte para cámaras Firewire y tarjetas de televisión
  - Soporte para tasas de cuadros variables
  - Soporte de formatos RGB24, YUVA, YUV, y jpg
  - Soporte de codificación mpeg1, mpeg2, h264, ogg, matroska y xvid.
- Audio
  - Soporte para mp3, mod, it, xm, y wav
  - Modificación del muestreo para audio
  - Permite grabar de cualquier fuente externa
  - Ajuste de velocidad y dirección de la reproducción

3.1.2.2.2 Sistema de gestión, almacenamiento, playout de código abierto

Como se mencionó anteriormente, la solución aconsejable para el manejo de gestión de contenido es VLC Media Player. Este paquete de software permite la emisión de contenidos de audio y video mediante su interfaz web lo cual nos facilita la administración de las bibliotecas digitales para broadcasting de contenido multimedia.

Figura 3.10. Video Lan Player de Video LAN Organization

Las principales características de VLC que lo hacen la mejor opción para un sistema de gestión de contenidos son las siguientes:

- Video:
  - Compatibilidad con MPEG1, MPEG2, MPEG3 Y MPEG4
  - Salida de flujos de transporte PS, ES, TS, PVA
  - Codecs para video MKV, AVI, MOV, WMA
- Audio:
  - Compatibilidad de ingreso de audio AAC, DTS, AC3/A52, y MIDI
- Ingreso multimedia:
  - Interfaz UDP/RTP Unicast y Multicast
  - Interfaz HTTP/FTP para gestión
  - Interfaz TCP Unicast para servidores
  - Ingreso de DVD, Video CD, DVB
- Salida multimedia
  - Formatos de salida nativos X11, XVideo, SDL y ASCII de video para tarjetas moduladoras
  - Formatos de salida multicanal de audio multicanal analógicos y digitales

3.1.2.2.3 Sistema de multiplexor de contenido de código abierto

OpenCaster, como se mencionó en capítulos anteriores, provee las siguientes características, que serán aprovechadas en el presente proyecto:

- Soporte para audio DTS
- Soporte para Phyton y Debian
- Conversión de archivos .srt a subtítulos por teletexto
- Soporte para codificación de audio h264
- Aplicabilidad para 24fps (frames per second)
- Optimización de compresión para archivos de carrousel
- Soporte para Ginga y HbbTV
- Generación de descriptores de audio y video
- Remultiplexación
- PlayOut para transporte de MPEG2
- Driver para IPTV (multicast)
- Integración con módulos de Cisco, Ericsson, Eurotek, Harmonic, Mitan, Wellav y Adtec

Con lo que en general se puede lograr utilizando OpenCaster:

- Creación de tablas PSI/SI (tablas de transmisión de contenidos)

- Solución de PlayOut para datos, audio y video codificado
- Manipulación en tiempo real del stream de transporte (TS)

3.2 Gestión de archivos multimedia

A continuación se realizará la automatización del proceso de transmisión de un canal digital utilizando VLC Media Player como una opción de software libre. El uso de VLC como se explicó anteriormente permite que el ordenador funcione como un servidor unicast o multicast de audio y video. Esta configuración se realiza utilizando la funcionalidad VideoLan Manager o VLM, la cual permite crear la secuencia de reproducción de videos.

3.2.1 Programación en VLC [20]

VLC se encarga de hacer un streaming (flujo) de video a una dirección específica. Una vez descargada e instalada la aplicación el procedimiento para crear una interfaz de flujo compatible con el sistema brasileño es la siguiente:

1.- Seleccionamos la opción de Emitir dentro del menú Medio

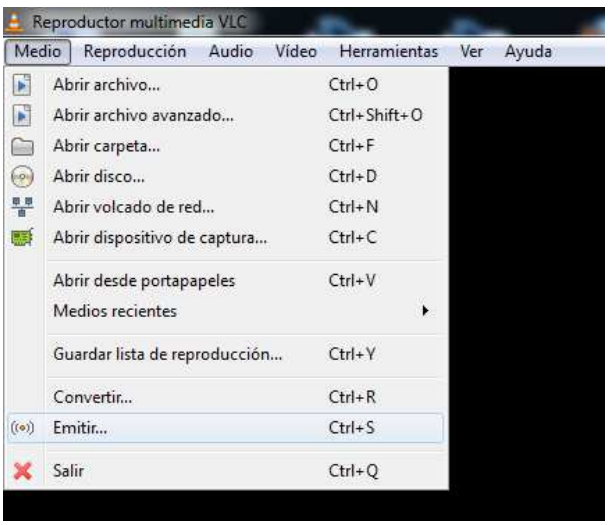


Figura 3.11. Opción “Emitir” en VLC

2. Añadimos el archivo, en nuestro caso el video que se transmitirá y hacemos click en Emitir.

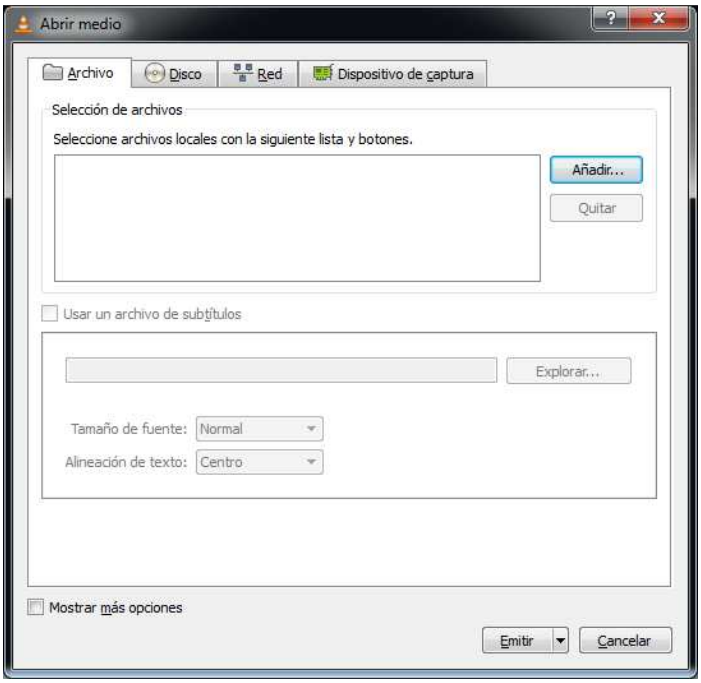


Figura 3.12. Navegador de medios de VLC

3. Configuramos la salida de emisión, en la cual debemos crear la interfaz UDP con la dirección y el puerto que utilizaremos.

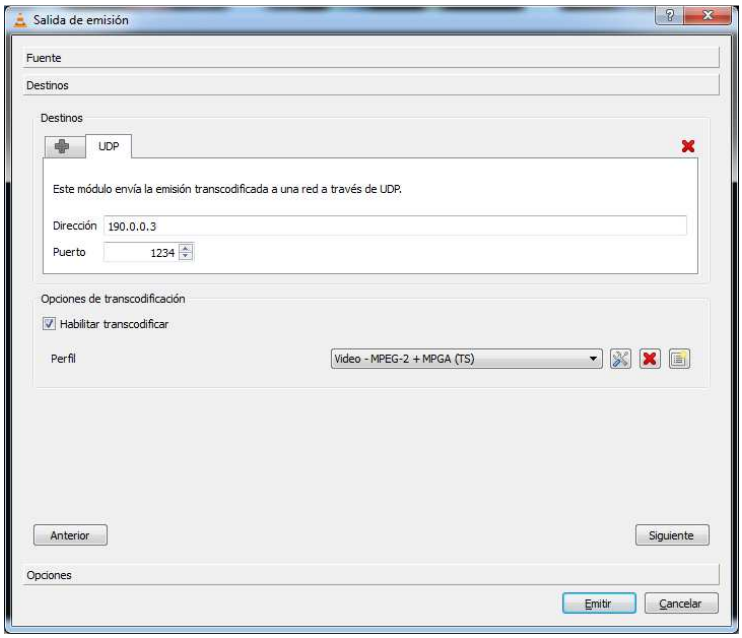
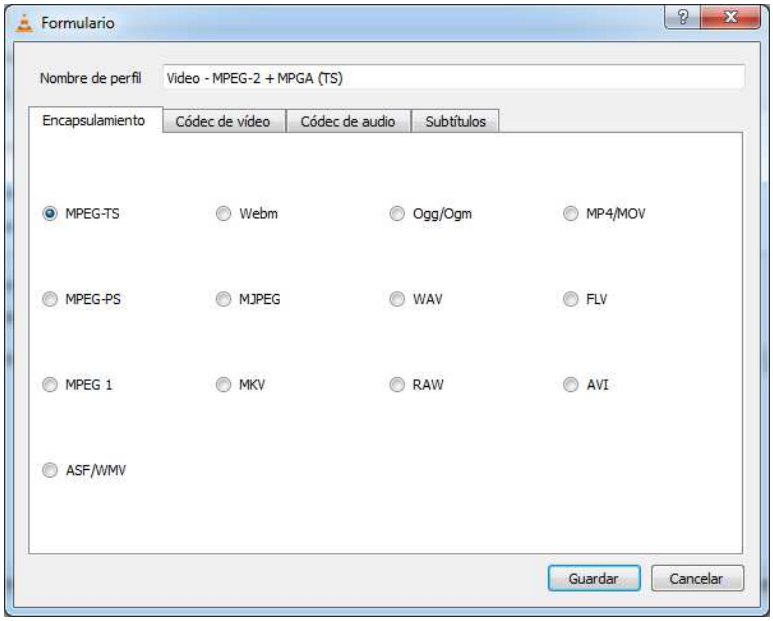
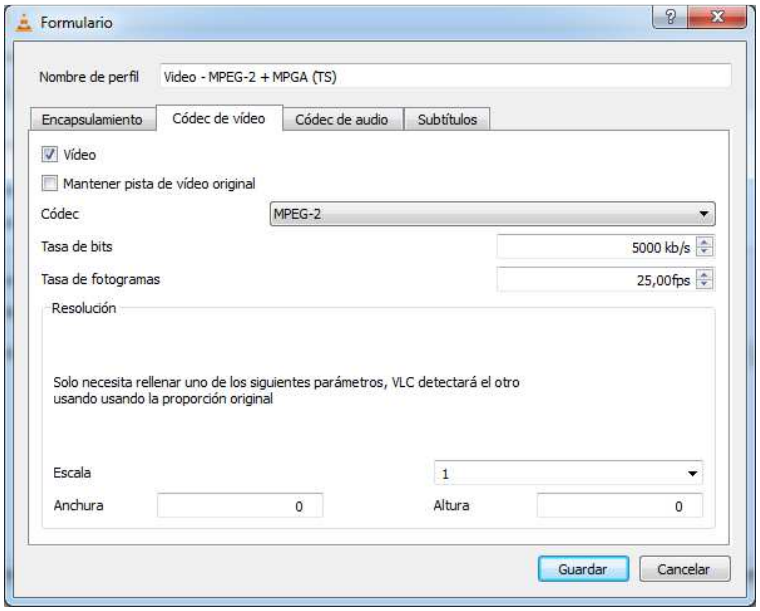


Figura 3.13. Configuración de la salida de emisión en VLC

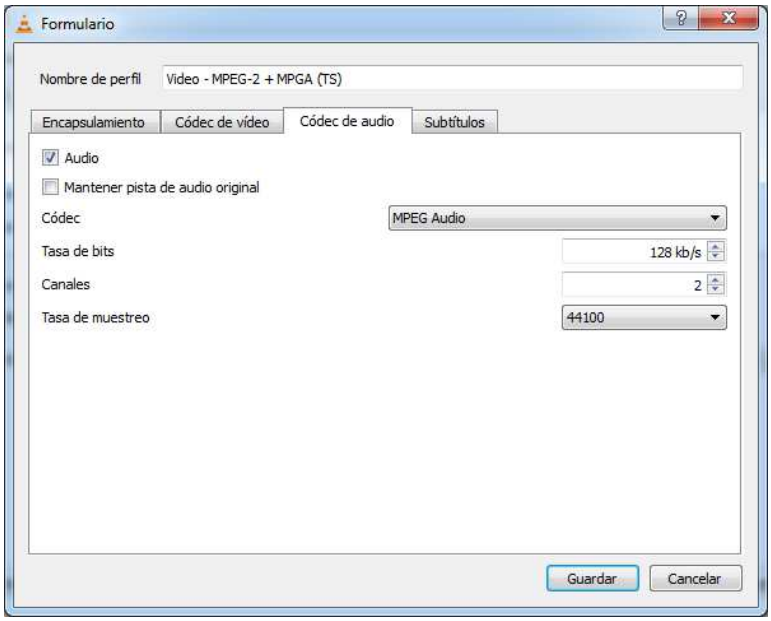
4. Configuramos las opciones de transcodificación, es decir, especificamos el tipo de encapsulamiento, el códec de audio y video, y la opción de subtítulos



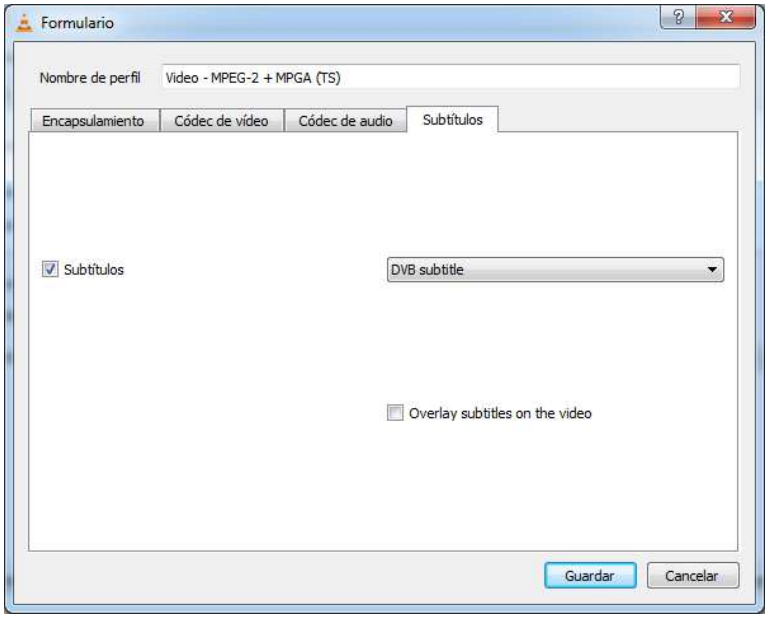
(a)



(b)



(c)



(d)

Figura 3.14. Perfil de emisión en VLC (a) Encapsulamiento (b) Códec de video (c) Códec de audio (d) Opción de subtítulos

5. Luego de guardar los cambios, configuramos las preferencias del muxor, es decir la configuración de los paquetes para el TS.

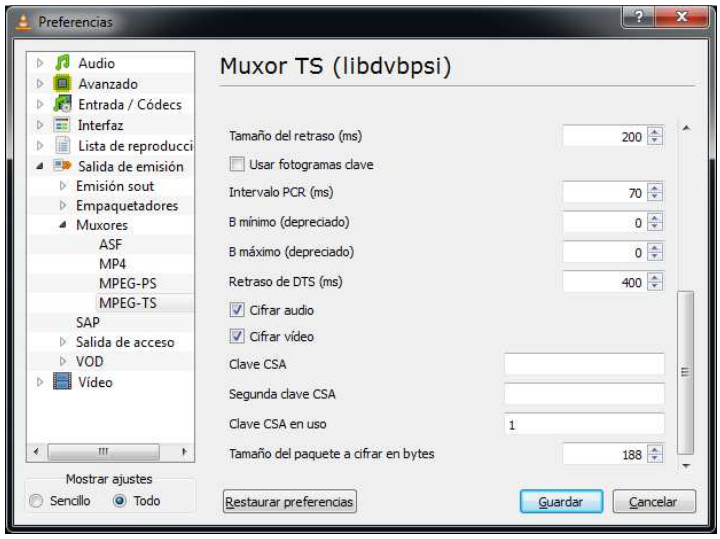


Figura 3.15. Preferencias del multiplexor del Transport Stream

Una vez configurado el VLC, se puede hacer la recopilación del stream con OpenCaster en cualquier ordenador de la red. Para automatizar este procedimiento se utilizará la aplicación VLM dentro de VLC.

3.2.2 Configuración de VLM [2]

VLM es una interfaz de instrucciones para la automatización de VLC, y la administración de contenidos. Para esto se utilizan dos interfaces: La interfaz Telnet, y la interfaz web (http). La segunda opción es la más fácil de configurar, por lo cual se utilizará la misma en este proyecto.

Para que VLC se pueda configurar mediante VLM, se debe crear un archivo de texto con los comandos a ejecutar. La estructura de un archivo de configuración en VLM se basa en elementos, los cuales pueden ser de tres tipos:

1. Elementos broadcast.- los cuales definen las propiedades de los medios de emisión o videos a transmitir
2. Elementos de programación.- Que permiten relacionar temporalmente la ejecución de los elementos de broadcast.
3. Comandos de control.- Los cuales definen el comportamiento de los elementos de broadcast (inicio, parada, pausa, etc.)

El archivo VLM se estructura con los siguientes comandos básicos [2]:

- **New .-** crea un elemento nuevo de video. Se debe especificar si es de video bajo demanda (VOD), un elemento broadcast o una programación de contenido (Schedule)
- **Setup.-** Designa las propiedades del elemento creado. Puede utilizarse para borrarlo, o ejecutarlo.
- **Control.-** se utiliza para cambiar el estado de reproducción de un elemento
- **Input.-** especifica el archivo de ingreso para la reproducción
- **Output.-** define la interfaz de salida de la emisión
- **Enabled/Disabled.-** habilita o deshabilita el elemento creado
- **Loop.-** activa la repetición de un elemento luego de que finalice
- **Append.-** añade un comando al archivo VLM
- **Date.-** permite crear una fecha para la reproducción y finalización de un elemento. Debe estar en el formato "año/mes/día-hora:minutos:segundos"
- **Period.-** especifica un periodo de tiempo en el cual la programación se vuelve a ejecutar
- **Repeat.-** Designa el número de veces en la que una emisión se repetirá.
- **Play.-** Inicio de una emisión
- **Pause.-** Pausa de la emisión
- **Stop.-** Fin de la emisión
- **Seek.-** inicio de la emisión en un tiempo específico del elemento de video

Con estos comandos, el archivo VLM utilizado tiene la siguiente estructura:

```
New transmision1 broadcast enabled loop

Setup transmision1 input "C:\video.mp4"

Setup transmision1 output

#transcode{vcodec=mp2v, vb=4096, scale=1, acodec=mpga, ab=128, channels=2}:
std{acces=udp, mux=ts, dst=192.0.0.1:1234}
```

Para la administración de VLM utilizamos la interfaz web de VLC, la cual se activa como se muestra en la figura 3.16:

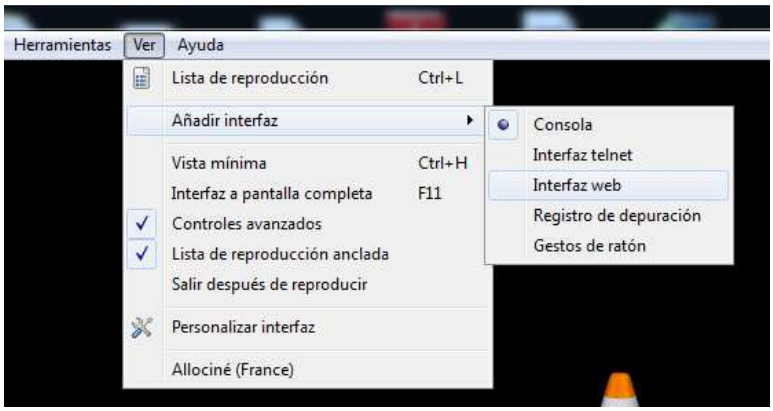


Figura 3.16. Añadir una interfaz para VLM

Luego de activar la interfaz web, podemos acceder a la misma en cualquier navegador de internet, entrando a la dirección http://localhost:8080/vlm.html o http://127.0.0.1:8080/vlm/html . La página de configuración a la cual se accede se muestra en la figura 3.17:

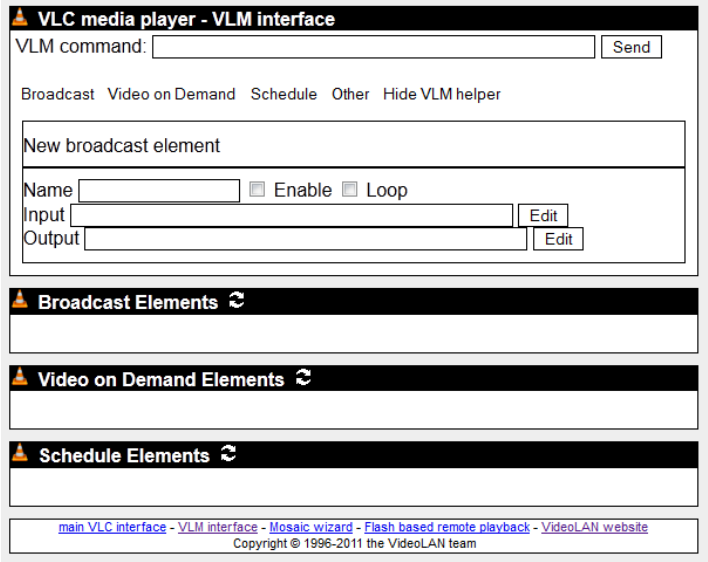


Figura 3.17. Página de acceso a VLM

Una vez ingresados a la ventana de configuración, agregaremos un elemento de broadcast para la transmisión. Primero se le da el nombre que utilizaremos y se activan los atributos enable, y loop para asegurar su reproducción, como se puede ver en la figura 3.18:

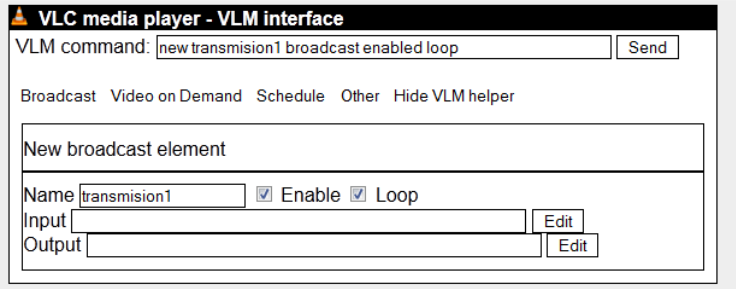


Figura 3.18. Ingreso de un elemento de broadcast en VLM

Luego agregamos el elemento multimedia para emitir, esto se hace haciendo click en el botón Edit del campo Input (figura 3.19).

Figura 3.19. Navegador de archivos de VLM

Aceptamos, e ingresamos al campo Output para definir la salida. En la figura 3.20 se muestra la ventana de configuración, en la cual ingresamos la dirección IP, el puerto UDP, y los parámetros de codificación.

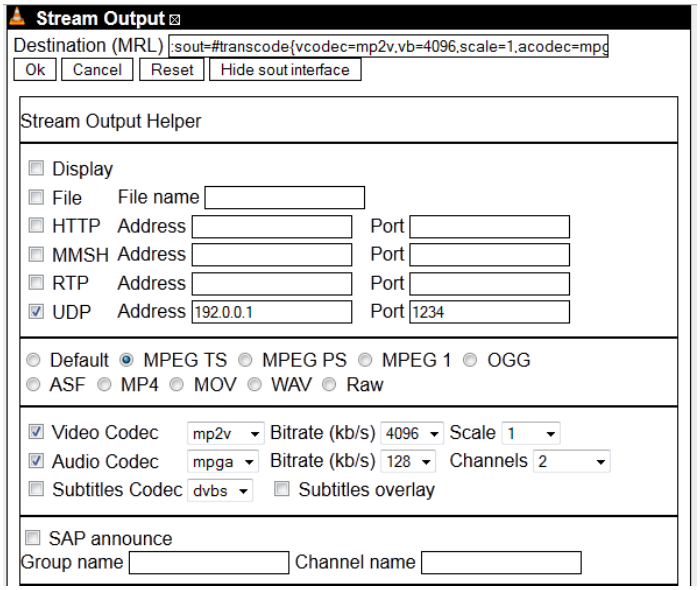


Figura 3.20. Configuración de la salida de streaming para VLM

Por último hacemos click en “Send” para guardar los cambios en la máquina VLM. Si no existen errores, la ventana que se observará será la de la figura 3.21.

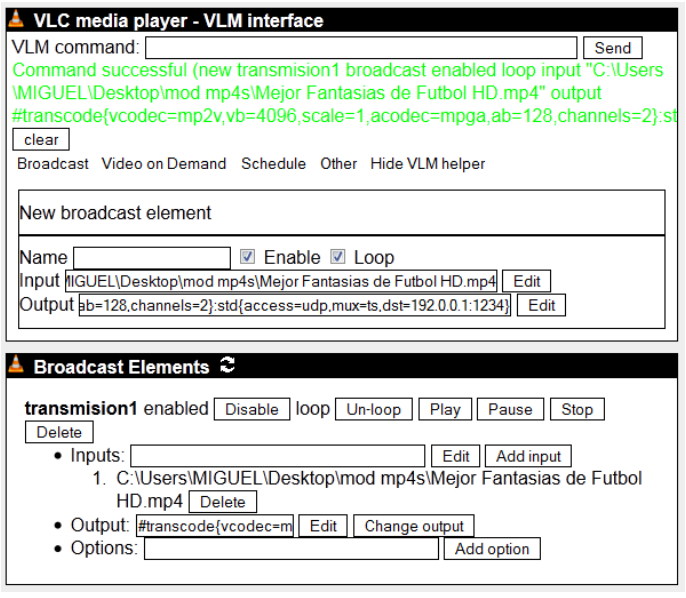


Figura 3.21. Pantalla de VLM con un elemento de broadcast agregado satisfactoriamente

El archivo VLM generado con la interfaz web, se muestra a continuación:

```
##

##
## VLM HTTP interface
## This file can be loaded as is in VLM.
## Comments starting with "##" were added by the HTTP interface.
## You can remove them if you want to.
##

# VLC media player VLM command batch
# http://www.videolan.org/vlc/

new transmision1 broadcast enabled loop
setup transmision1 input "C:\Users\MIGUEL\Desktop\mod mp4s\Mejor Fantasias de Futbol HD.mp4"
setup transmision1 output #transcode{vcodec=mp2v,vb=4096,scale=1,acodec=mpga,ab=128,channels=2}:std{access=udp,mux=ts,dst=192.0.0.1:1234}

##
## end of export
##
```

3.2.3 Configuración de OpenCaster

OpenCaster es una plataforma libre, que para la presente aplicación correrá en un sistema Linux. Se generará el Flujo de Transporte en formato MPEG2 con lo que se puede almacenar dicho archivo en memoria, enviarlo a través de una red multicast por una dirección IP, o se lo puede modular con una tarjeta específica. Debido a que el alcance de esta tesis está enfocado al software requerido, se utilizará la interfaz IP para la generación del archivo TS.

El objetivo perseguido con este paquete de software es la adaptación de los Flujos de Transporte mediante la multiplexación de audio y video, así también como la inclusión de una aplicación NCL al canal de datos.

3.2.3.1 Instalación de OpenCaster y sus dependencias [20] [25]

Para instalar OpenCaster, se tomará en cuenta que ya se tiene configurado correctamente una distribución de Ubuntu dentro en el computador, con lo que se deben seguir los siguientes pasos:

Paso 1

Descargar los archivos de instalación de www.avalpa.com. Este procedimiento se hace mediante la creación de una cuenta con avalpa, pero los documentos son gratuitos.

Con el gestor de paquetes Synaptic, instalar las siguientes dependencias y sus respectivas subdependencias:

- Gcc 4.3
- Binutils
- Libc6-dev
- Libgomp1
- Linux-libc-dev
- Make
- Python 2.5
- Libpcap0.8
- Zlib1g-dev

Paso 2

Descargar el parche para televisión digital con Ginga desarrollado por Lifia, de ftp://tvd.lifia.info.unlp.edu.ar/OpenCaster2.4/.

Luego creamos una carpeta en la cual se instalará el programa, en este caso /home/user /tvd/ OpenCaster. A continuación descomprimimos el archivo con el siguiente comando:

```
$ cd /home/user/tvd/OpenCaster

$ tar zxf OpenCaster.2.4.tgz
```

Copiamos el parche de Lifia a la carpeta descomprimida y lo aplicamos.

```
$ cd OpenCaster2.4

$ patch –p 1 < ../OpenCaster2.4-lifia-rev362.patch
```

Por último con privilegios de administrador compilamos

```
#make

#make install
```

Con lo cual tendremos instalado satisfactoriamente OpenCaster.

3.2.3.2 Librerías de OpenCaster y multiplexación del TS [20]

OpenCaster debe ser configurado de manera que cumpla con la norma brasileña, para lo cual se debe especificar los siguientes aspectos dentro de su configuración:

- Los elementos de audio y video a transmitir
- La información de las tablas PSI-SI las cuales definen los parámetros estructurales del archivo TS.
- La aplicación interactiva que se ejecutará en el transmisor

Se debe tener en cuenta que el TS debe tener una tasa de bits constante, por lo que se debe rellenar con un flujo de paquetes con información redundante o en el peor de los casos utilizando paquetes vacíos.

Los parámetros de las tablas PSI que se deben especificar son los siguientes [2] [7]:

- **PAT (Program Asociation Table) .-** Indica los PID's (Program Identificator) de las tablas PMT de cada servicio anexado.
- **PMT (Program Map Table) .-** Indica la asociación de un servicio y el tipo de contenido enviado en el TS.
- **NIT (Network Information Table).-** Indica la red en la cual se envía o se recibe el TS.
- **SDT (Service Description Table).-** Describe parámetros generales sobre el servicio de televisión que se va a multiplexar, como nombre del servicio, proveedor, categorización, etc.
- **EIT (Event Information Table).-** Adjunta parámetros temporales sobre la reproducción del TS, y además lleva información sobre la guía de programación
- **AIT (Aplication Information Table)**.- Describe las aplicaciones interactivas que se envían en el TS.

Para la generación de un archivo que tenga estos parámetros se trabaja con Scripts en Python. Python es un lenguaje interpretativo, al igual que Lua, con el cual se pueden crear los TS que tienen las tablas PSI y su modificación en tiempo real. Para este objetivo también se pueden utilizar los comandos propios de OpenCaster con los

cuales se pueden multiplexar varios TS en uno solo, con lo que se logra una transmisión de varios canales digitales.

Un documento o script en python tiene cuatro partes principales para su estructuración, las cuales son:

1. Llamado a librerías de OpenCaster, y definiciones de clases
2. Variables de entorno para edición del TS
3. Definición de parámetros para cada una de las tablas de multiplexación
4. Instrucciones para la creación de las tablas PSI-SI

Librerías

Lo primero que debemos hacer es crear un archivo de texto el cual contiene el script en python. En este caso será un archivo con la dirección /home/tvd/servicio1/gtables.py . El encabezado de librerías se define de la siguiente manera:

```
#!/usr/bin/env python

import os

from dvbobjects.PSI.PAT import *

from dvbobjects.PSI.NIT import *

from dvbobjects.PSI.SDT import *

from dvbobjects.PSI.PMT import *

from dvbobjects.SBTVD.Descriptors import *
```

Variables

Para que el script tenga una configuración universal, se utilizan variables con las cuales se puede modificar el servicio de televisión digital en OpenCaster.

```
tvd_ts_id = 1 # ID de red.

tvd_orig_network_id = 1 # ID de red original.

ts_freq = 533 # Frecuencia de transmisión

ts_remote_control_key = 0x05 # Tecla de control remoto.

tvd_service_id_sd = 1 # ID de servicio de TV Digital.
```

```
tvd_pmt_pid_sd = 1031 # PID de la PMT del servicio.
```

Parámetros de tablas

Los parámetros para las tablas generalmente se trabajan en forma de clases como en Java. Las clases se dividen en secciones, las cuales describen el contenido de las tablas PSI. El código utilizado se muestra en el anexo. Entre las clases más importantes se destacan

- System_management_descriptor: en la cual se define las propiedades des sistema ISDB en cuanto a información sobre qué es el sistema y qué está transmitiendo.
- Terrestrial_reception_descriptor: en la cual se definen parámetros de modulación
- Transport_stream_information_descriptor: que define propiedades del TS como nombre, tecla del control remoto, información del servicio,etc.
- Partial_reception_descriptor: clase para la utilización de servicios 1-seg (móviles)
- Service_description_section: contenido de la tabla SDT la cual se utiliza para especificar los servicios disponibles dentro del transport stream.
- Program_association_section: clase con el contenido de la tabla PAT, la cual mapea los servicios PMT
- Program_map_section: contiene la descripción de la tabla MPT, la cual define los flujos de datos. En esta clase debe especificarse el stream de audio y video.

Instrucciones para la creación de tablas y el TS

El código utilizado se puede observar en el anexo. Las sentencias utilizadas son parte de los fuentes necesarios para generar el multiplexado del flujo de transporte. En este apartado se abren archivos de configuración y se escriben las tablas en archivos con extensión ".ts".

Una vez creada las tablas el archivo en python debe ejecutarse con los siguientes comandos:

```
$ cd/home/user/tvd/servicio1/

$ chmod u+x gtables.py

$ ./gtables.py
```

Por último se ejecuta el multiplexado en OpenCaster de la siguiente manera:

```
$ cd /home/user/tvd/servicio1

$ cp /home/user/tvd/OpenCaster/OpenCaster2.4//video.ts .

$ cp /home/user/tvd/OpenCaster/OpenCaster2.4/audio.ts .

$ cp /home/user/tvd/OpenCaster/OpenCaster2.4/null.ts .

$ tscbrmuxer \
        600000 \
        b:15040 pat.ts \
        b:15040 pmt_sd.ts \
        b:3008 sdt.ts \
        b:3008 nit.ts \
        b:2300000 firstvideo.ts \
        b:188000 firstaudio.ts \
        b:27434198 null.ts > prueba1.ts
$tsstamp transmision1.ts 29958294 > transmisión 1.fixed.ts
```

donde:

- 600000 es la cantidad de paquetes que se van a multiplexar
- 15040: ancho de banda para el envío de las tablas PAT y PMT
- 2300000 y 188000: corresponden al ancho de banda de audio y video respectivamente.
- 27434198: Es el ancho de banda de los paquetes de relleno. Se debe tener en cuenta que el ancho de banda total de un TS para el sistema ISDB-T es de 29958294 bps.

Con esta configuración, se obtiene un flujo de transporte de audio y video listo para transmitir, a lo cual debemos agregar la aplicación NCL que queremos que se ejecute. Para esto debemos seguir los siguientes pasos:

Generación del carrousel de objetos con la aplicación

Se crea una carpeta con la aplicación en un directorio por defecto, en el cual OpenCaster realizará la búsqueda del archivo y lo multiplexará con el audio y video. Los comandos utilizados son los siguientes:

```
$ cd /home/user/tvd/servicio1

$ mkdir app_ginga

$ cp -r <path a la aplicación>/* app_ginga/

$ oc-update.sh app_ginga 0x0C 1 2004 2
```

Donde:

- App_ginga: es el directorio por defecto para el almacenamiento de aplicaciones
- 0x0C: association_tag del carrousel.
- 1: es la versión del módulo generado
- 2004: PID en el que se envía el Carrousel
- 2: ID del carrousel

Generación de las secciones de la tabla AIT con la información de la aplicación enviada

El código utilizado se muestra en el anexo. Destacan las siguientes clases que se agregan para adjuntar la aplicación interactiva:

- Application_information_section: clase en la cual se describen los parámetros de la aplicación, como el tipo, el id de la aplicación, etc.
- Application_signalling_descriptor: describe parámetros como la versión de la tabla AIT, y el tipo de aplicación anidada.

Multiplexación del transport stream con los nuevos datos agregados

Como se procedió anteriormente el multiplexado se realiza de la siguiente manera:

```
$ tscbrmuxer \

        600000 \

        b:15040 pat.ts \

        b:15040 pmt_sd.ts \

        b:3008 sdt.ts \

        b:3008 nit.ts \

        b:3008 ait.ts \
```

```
        b:400000 app_ginga.ts \

        b:2300000 firstvideo.ts \

        b:188000 firstaudio.ts \

        b:27031190 null.ts > prueba1.ts

$ tsstamp transmision1.ts 29958294 > transmision1.fixed.ts
```

Con todos estos pasos, el flujo de transporte está listo para ser transmitido, lo cual se puede hacer por una tarjeta moduladora, o por Ethernet como se mencionó en subcapítulos anteriores.

3.3 Plataforma de contenido digital

Una vez especificado el aspecto de transmisión sobre la red creada entre OpenCaster y VLM, se procederá con el diseño de la aplicación interactiva que se ejecutará mientras se reproduce el video.

Como en este caso, diseñaremos una aplicación que se ejecutará durante la transmisión de un partido de fútbol, debemos considerar los siguientes aspectos:

- La aplicación se reproduce desde el inicio hasta el fin del partido, es decir, no cambia durante la transmisión, pero debe ser capaz de soportar estos cambios.
- La aplicación debe proporcionar información completa sobre el evento transmitido, tal es el caso de formación de los equipos, tabla de posiciones, partidos en la fecha, etc.
- El espacio que utiliza la aplicación en la pantalla debe ser mínimo, pues el interés del espectador está sobre el elemento de video, la opción de interactividad debe ocultarse en cualquier momento.
- La navegación por los menús debe ser intuitiva y amigable

Con estos parámetros básicos, se diseñará la aplicación para un partido de fútbol en el cual se mostrará el video como aspecto principal.

3.3.1 Diseño NCM

El diseño de aplicaciones interactivas, como se explicó anteriormente, se basa en el modelo NCM, el cual es una aproximación a una máquina de estados. La aplicación

diseñada para el presente proyecto tiene el modelo NCM que se muestra en el anexo 2.

El proceso para realizar un modelamiento utilizando el lenguaje NCM, se basa en los estados en los que la aplicación espera por los comandos respectivos, los cuales indican a cuál será el estado siguiente.

La aplicación realizada utiliza estados en la máquina NCM, en cada uno se especifica:

- Identificador del estado
- Ingresos y salidas para estados futuros o anteriores
- Acción en la cual se pasará al siguiente estado
- Puertos de interacción

El identificador de estado por defecto junto con sus ingresos y salidas en el Lenguaje NCM se muestra en la siguiente figura 3.22:

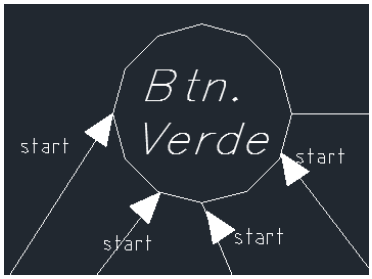


Figura 3.22. Identificador de estado con entradas y salidas

Los puertos de interacción son estados intermedios, que permiten graficar el comportamiento de los estados NCM. Se los describe como se muestra en la figura 3.23:

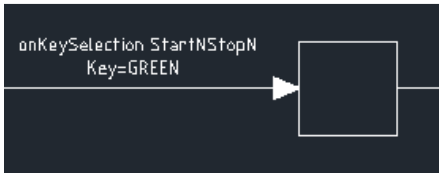
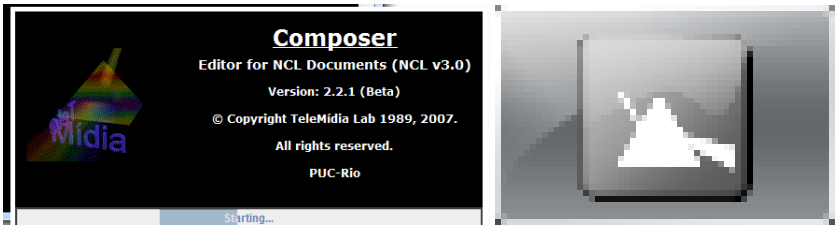


Figura 3.23. Puerto de interacción con los estados

3.3.2 Diseño de la aplicación NCL-Lua

Una vez diseñado el diagrama NCM para la aplicación, debemos trasladarlo a lenguaje NCL, y a los Scripts necesarios en Lua. Cabe recalcar que una aplicación de este tipo puede realizarse íntegramente en NCL sin el requerimiento de un script Lua que se ejecute, pero como se utilizará una máquina virtual de Ginga, la cual debe "interpretar" la interacción con el teclado de una computadora y no con la señal infrarroja de un control remoto como sería normalmente, el script en Lua es obligatorio.

Para trasladar el diagrama NCM a NCL se utiliza la herramienta Composer, el cual permite crear aplicaciones interactivas basándose en el diseño NCL. Composer es una herramienta gratuita proporcionada por la PucRío.



(a) (b)

Figura 3.24. Herramienta Composer (a) Inicialización de la aplicación (b) Ícono de Composer NCL

El entorno de programación de Composer es intuitivo, y permite tener un control total sobre el tiempo de reproducción y la manera en que queremos que se ejecuten los elementos multimedia. En la figura 3.25 se observa la pantalla principal de programación:

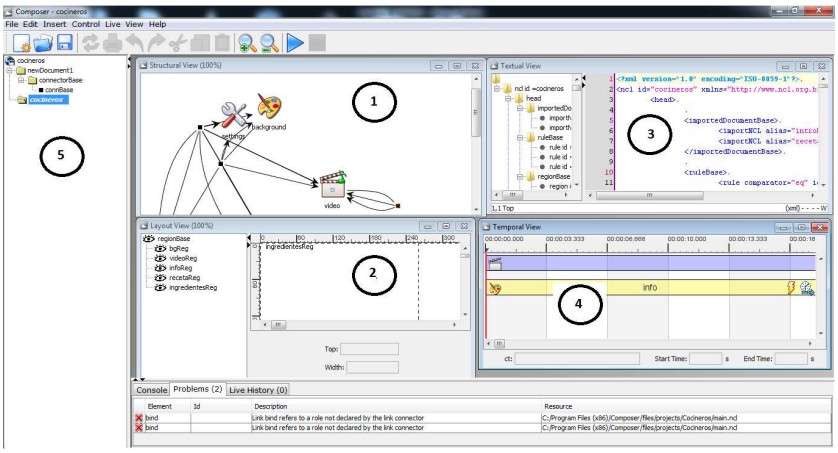


Figura 3.25. Pantalla principal de Composer

En la figura se pueden diferenciar cinco partes principales, las cuales son:

1. Vista estructural.- Define la estructuración (dinámica) del documento NCM
2. Vista de capas.- Define cada una de las capas donde se ejecutarán los elementos multimedia. Representa porciones de la pantalla del televisor.
3. Vista textual.- En esta ventana se observa el archivo NCL generado
4. Vista temporal.- Representa el tiempo de ejecución de los elementos multimedia dentro del archivo NCL
5. Navegador de proyecto.- Aquí se observan todos los documentos agregados a un proyecto en Composer.

Para crear un proyecto en Composer se procede de la siguiente manera:

Paso 1

Creamos un proyecto nuevo en el menú “File/new/project”

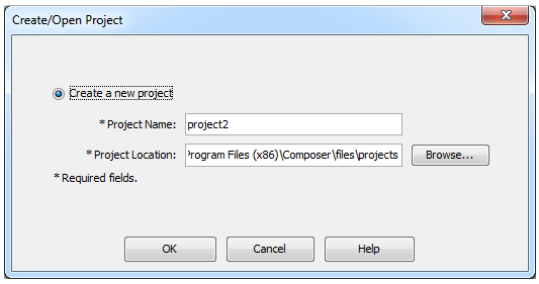


Figura 3.26. Creación de un nuevo proyecto en Composer

Paso 2

En el menú “Insert/Region” definimos la región en la que vamos a presentar nuestra aplicación

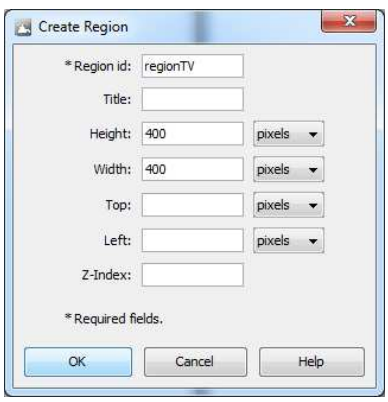


Figura 3.27. Creación de una región

Los parámetros de largo y ancho se pueden especificar en pixeles o en porcentajes de pantalla, además se los puede modificar en la ventana de “Vista de capa”, donde podemos mover los márgenes y la posición de la región, como se ve en la figura.

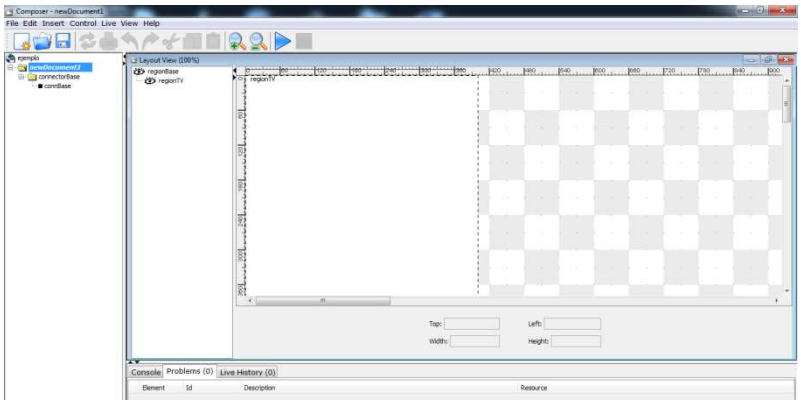


Figura 3.28. Vista por capas

Paso 3

En el menú “Insert/media node”, definimos el nodo multimedia que queremos reproducir, para esto debemos especificar la ubicación y el tipo de archivo. En el caso de archivos de video es muy importante especificar la extensión correcta del documento, pues si este parámetro es incorrecto, no se logrará reproducir el archivo NCL.

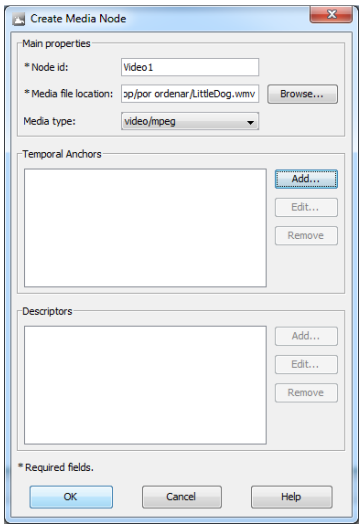


Figura 3.29. Creación de un nodo multimedia

Paso 4

Agregamos un descriptor al elemento multimedia. En la ventana de creación de descriptores debemos definir la región en la que se va a reproducir el archivo, también podemos especificar con qué reproductor se ejecutará, y la duración del archivo.

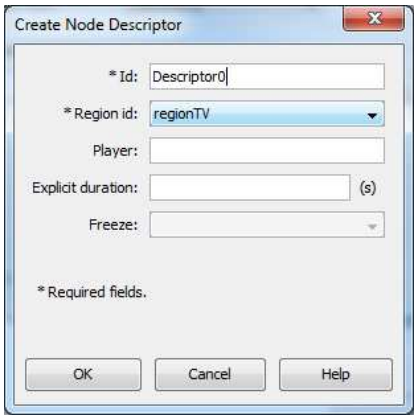


Figura 3.30. Creación de un descriptor asociado a una región

Paso 5

Luego de crear los elementos multimedia necesarios, creamos las conexiones respectivas entre los archivos, y creamos sus dependencias. Para esto seleccionamos los elementos que intervendrán y en el menú "Insert/link" creamos las dependencias necesarias.

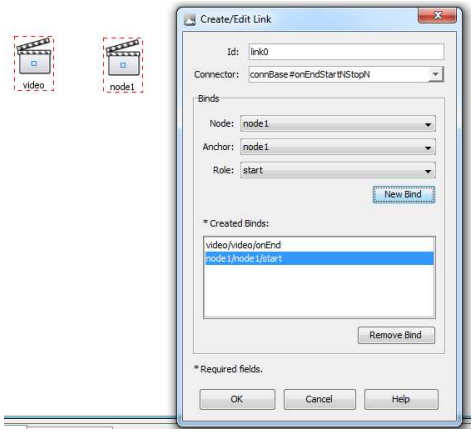


Figura 3.31. Creación de un link entre dos nodos multimedia

Una vez creado correctamente la conexión, podemos observar la composición del archivo NCM como se muestra en la figura 3.32:

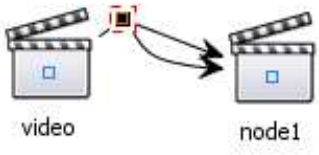


Figura 3.32. Nodos multimedia correctamente enlazados.

Siguiendo los pasos anteriores, se trasladó el modelo NCM a Composer, con lo que se obtuvo el diagrama estructural de la figura 3.33

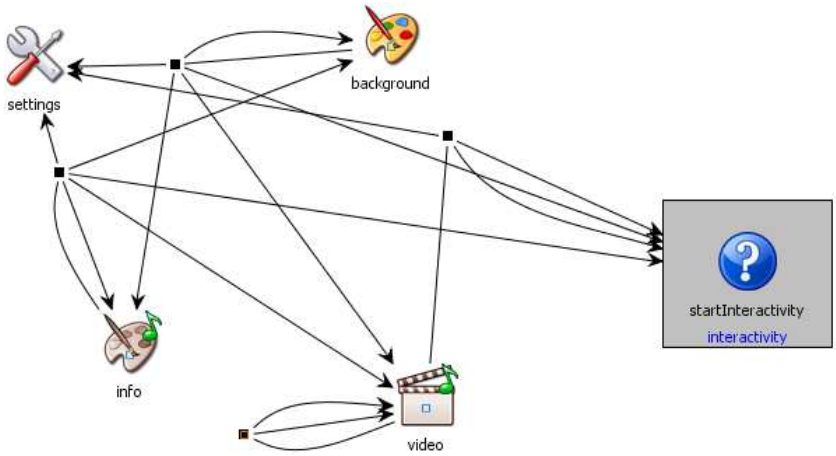


Figura 3.33 Diagrama estructural de la aplicación “DeportesAbierto.ncl”

El contexto de interactividad posee el control para la visualización del menú, y permite a la máquina virtual detectar el botón que fue presionado. Una visión más amplia del contenido del contexto de interactividad se muestra en la figura 3.34.

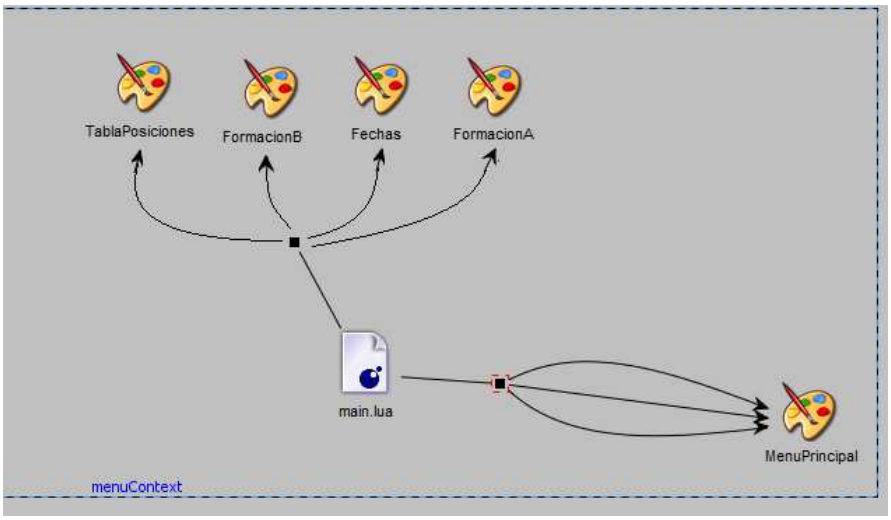


Figura 3.34. Contexto de interactividad de la aplicación “DeportesAbierto.ncl”

Como se puede observar, el contexto contiene la aplicación main.lua, la cual se encarga de acceder a los puertos de interacción con las diferentes imágenes que conforman el menú principal de la aplicación. Los nodos permiten acceder a la ejecución de los archivos multimedia, los cuales se mostrarán en diferentes partes de la pantalla, minimizando la pantalla principal con lo cual se asegura que los objetos no intervengan con la transmisión. En la figura 3.35 se muestra la disposición de capas para la aplicación diseñada.

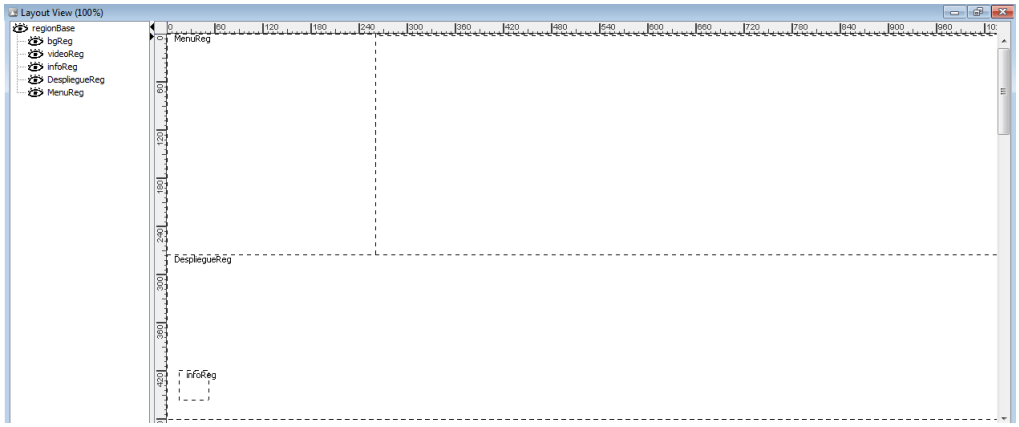


Figura 3.35. Disposición de capas para la aplicación “DeportesAbierto.ncl”

Una vez compilado y exportado el documento “ncl” a la máquina virtual de Ginga, se puede ejecutar la aplicación. En la figura 3.36 se muestra la pantalla principal, donde únicamente se puede observar el botón de información.

Figura 3.36. Pantalla principal

El botón de información indica que la interacción con el televisor está activada. Para acceder al menú de interactividad se debe presionar el botón de “Enter” en el control remoto del televisor, o en el caso de la máquina virtual, la tecla “Enter” del teclado. El funcionamiento programado radica en minimizar la pantalla principal, y ubicarla en la parte superior izquierda del televisor para que el menú disponible se ubique en el espacio restante. El menú principal se puede observar en la figura 3.37.

Figura 3.37. Menú principal de la aplicación

El menú indica la tabla de posiciones del campeonato, la formación del equipo local y visitante, y los partidos de la fecha. El menú se basa en mostrar un conjunto de imágenes previamente diseñadas y depuradas en un software para el efecto, en nuestro caso se utilizó Adobe Illustrator, pero se puede usar cualquier paquete de software libre expuesto en capítulos anteriores.

Capítulo IV:

DISEÑO DEL SOFTWARE PARA UN CANAL DE TELEVISIÓN DIGITAL DE DEPORTES CON RETORNO

4.1 Análisis de los requerimientos de un canal de deportes utilizando retorno por Ethernet

En el contexto de una aplicación para un canal de deportes, el canal de retorno juega un papel importante en concursos, votaciones (por ejemplo para elegir al mejor jugador de un encuentro), etc., por lo cual las aplicaciones interactivas pueden utilizar esta plataforma para comunicarse de una manera diferente con el usuario. Así el televidente tiene la opción de una verdadera interacción con la programación de un cierto canal, elevando el rating de un programa determinado.

Para este propósito, se diseñará una aplicación que permita al usuario la interacción con una programación en la cual podrá elegir entre opciones. Esta selección se enviará por internet a un servidor, en el cual se contabilizarán los votos y luego estos se los mostrará en pantalla.

Así también como en el caso de un canal abierto, la aplicación de interactividad debe cumplir los siguientes aspectos:

- Debe ser lo más pequeña posible para que pueda ser enviada por el carrousel de objetos
- Su diseño debe ser tal, que sea de fácil navegación por parte del usuario
- Debe tener un diseño atractivo, teniendo siempre en cuenta que en televisión lo principal es vender el contenido y no la aplicación
- Evitar que la aplicación ocupe la mayor parte del área de la pantalla
- Lograr que la interacción con el usuario sea lo más transparente posible, es decir, que el mismo no debe preocuparse de aspectos como su dirección IP, ni el puerto de comunicación, simplemente asegurarse que su set-top-box esté conectado a internet por su proveedor de servicios.

4.1.1 La clase TCP Lua

Una alternativa para el manejo de canal de retorno es el uso de la clase TCP-Lua. Esta librería permite hacer una conexión TCP entre el usuario y la emisora de televisión, lo cual facilita la interacción en aplicaciones t-goberment, en programas de concursos y la utilización del televisor como un navegador web.

La clase TCP Lua es una librería que permite enviar solicitudes TCP a un servidor remoto, permitiendo al programador utilizar las siguientes funciones en su aplicación:

- tcp.connect.- establece la conexión tcp a un dominio. Se debe especificar la dirección URL y el puerto por el que se realizará la comunicación (en aplicaciones web, lo más común es el puerto 80)
- tcp.send.- permite enviar un paquete de información tcp
- tcp.receive.- recibe el paquete tcp solicitado
- tcp.disconnect.- termina la sesión tcp con el servidor.
- Tcp.execute.- es una función que debe agregarse por defecto. Llama a todas las librerías necesarias.
- Tcp.handler.- Traductor de eventos

La estructura básica implementada en [30], para la clase TCP Lua es la siguiente:

```
tcp.execute (

function ()

tcp.connect('www.urlaconectar.com', 80)

tcp.send('GET /search?hl=pt- PE&btnI&q='..evt.value..'\n')

local result = tcp.receive()

if result then

result = string.match(result, 'Location: http://(.-)\r?\n') or 'no resuelto'

else

local evt = {

class = 'ncl',

type = 'attribution',

name = 'result',

value = result,

}

evt.action = 'start'; event.post(evt)

evt.action = 'stop' ; event.post(evt)

tcp.disconnect()

end

)

end
```

Con esta configuración, se puede lograr la conexión de una aplicación a cualquier servidor utilizando TCP y Ethernet. Otra opción que se puede utilizar el LuaSocket [29].

Como se puede observar la clase TCP es una implementación en Lua, por lo cual se la debe llamar desde una aplicación principal, y utilizarla como una librería. Para esto se escribió un script en Lua que se encargue de cargar las variables necesarias. En el Anexo 1 se puede observar el código utilizado para la clase tcp.

4.1.2 Programación en PHP

Debido a que se requiere una administración de los datos obtenidos en la aplicación Lua, se debe instalar un servidor capaz de realizar el conteo de votos, correr un script para la actualización de los mismos, y también que publique un dominio web para conectividad entre el usuario y la empresa.

Los scripts utilizados en http se conocen como aplicaciones PHP (hypertext preprocessor). PHP es un lenguaje de scripting utilizado en el lado del servidor como una opción para diseño web de páginas dinámicas. Para esto, PHP puede ser embebido dentro de un documento HTML el cual será interpretado por un servidor web (para aplicaciones libres se utiliza Apache). PHP es un paquete de software libre, por lo cual disminuye los costos en el proceso de implementación.

Figura 4.1. Logotipo de PHP

Un script PHP puede realizar las siguientes acciones:

- Obtener información de una página web y realizar procesos como: grabarla en una base de datos, crear páginas condicionales dependientes del formato de la información, manejo de cookies, enviar e-mails, etc.
- Autentificación y seguimiento de usuarios
- Servidor de páginas para diferentes buscadores o dispositivos
- Servidor XML

- Manejo de conexiones TCP

Por lo tanto este lenguaje nos permitirá manejar las conexiones entrantes y salientes que se obtendrán con el documento creado en Lua para este propósito.

4.1.2.1 Estructura de un documento PHP y HTML

El intérprete de PHP sólo ejecuta las sentencias que se encuentran dentro de los delimitadores propios del documento. El código en sí se inserta dentro de una página HTML, el cual se escribe en texto plano. Un documento común de PHP dentro de una página HTML se muestra a continuación:

```
<!DOCTYPE html>
<html>                                  /*Encabezado html
    <head>
        <meta charset="utf-8" />
        <title>PHP Test</title>
    </head>
    <body>                              /*Cuerpo html
        <?php                           /*Script php
            echo 'Hello World';
        ?>
    </body>
</html>
```

Como se puede observar el documento sigue la estructura XML descrita en capítulos anteriores, por lo cual aseguramos una compatibilidad con los scripts en Lua.

PHP es un lenguaje muy amplio y es utilizado en casi todas las plataformas web que interactúan con bases de datos o con conexiones con el usuario, por lo cual a continuación se hará una revisión de la sintaxis básica de un documento. Posteriormente se explicarán las librerías, módulos y sintaxis utilizadas dentro del documento generado para el funcionamiento del presente proyecto.

4.1.2.2 Sintaxis básica de PHP

Para empezar a programar los scripts en PHP debemos tener una noción de la sintaxis básica que emplea este lenguaje. Debido a que sigue el modelo XML, y se lo puede embebir en HTML no es muy diferente a los otros lenguajes tratados en este proyecto, tal es el caso de Lua, Python y NCL. A continuación se describirán las sintaxis más utilizadas en PHP:

Nombre del archivo.- El nombre del script puede ser cualquiera, con extensión .php. Algunos archivos pueden ser .php3, .php5, o .phtml dependiendo del contenido del archivo plano.

Comentarios.- Para insertar un comentario en un documento PHP se procede de igual manera que en un archivo HTML, es decir se utiliza "//" para un comentario de una sola línea, o "/*" para iniciar un comentario multilínea y "*/" para cerrarlo. Por ejemplo:

```
<?php

// comentario de una línea

Print ("Hola mundo");

/* comentario de varias líneas

No se toman en cuenta por el

Compilador

*/

¿>
```

Delimitadores de código.- Un documento PHP empieza con "<?php" y termina con "?>", cada línea de comando termina con un ";"

Declaración de variables.- una variable en PHP se designa comúnmente con el signo "$" seguido del nombre de la misma, de la siguiente manera

```
<?php

$variable1= "Hola"

Print("$variable1 mundo")

?>
```

Arreglos.- Para trabajar con arreglos en PHP se puede proceder de dos formas. Primero se puede declarar una matriz normal entre paréntesis, y los elementos separados por comas de la siguiente manera:

```
$arreglo1=array(“elemento1”, “elemento2”, “elemento3”);
```

También se puede crear un arreglo asociativo utilizando la descripción de las columnas, es decir clasificando a los elementos siguiendo la siguiente estructura

```
$arreglo2=      array (
                        Nombre=> “Miguel”
                        Descripción=> “Estudiante”
                        Sexo=> “Masculino”
                        Edad=> “23”
                    );
```

Operador If-else-elseif.- Consiste en una pregunta causal “Si” como en cualquier lenguaje de programación previamente estudiado. Su sintaxis en PHP es la siguiente:

```
If (condición) {
//condición verdadera
} elseif{
// otra condición verdadera
} else {
// condición falsa
}
```

Comparaciones y operadores lógicos.- Para encontrar similitudes o comparar variables se utilizan los siguientes símbolos

Tabla 4.1. Comparadores

Ejemplo	Nombre	Resultado
$a == $b	Igual	TRUE si $a es igual a $b.
$a === $b	Idéntico	TRUE si $a es igual a $b, y son del mismo tipo.

$a != $b	No Igual	TRUE si $a no es igual a $b.
$a <> $b	No igual	TRUE si $a no es igual a $b.
$a !== $b	No idéntico	TRUE si $a no es igual a $b, o no son del mismo tipo.
$a < $b	Menor que	TRUE si $a es menor a $b.
$a > $b	Mayor que	TRUE si $a es mayor a $b.
$a <= $b	Menor o igual	TRUE si $a es menor o igual $b.
$a >= $b	Mayor o igual	TRUE si $a es mayor o igual $b.

Los operadores lógicos nos permiten realizar comparaciones dentro de operadores if o while

Tabla 4.2. Operadores lógicos

Ejemplo	Nombre	Resultado
$a and $b	And	TRUE si $a y $b son TRUE.
$a or $b	Or	TRUE si cualquiera, $a o $b son TRUE.
$a xor $b	Xor	TRUE si cualquiera $a o $b son TRUE, pero no ambos.
! $a	Not	TRUE si $a no es TRUE.
$a && $b	And	TRUE si $a y $b son TRUE.
$a || $b	Or	TRUE si cualquiera, $a o $b son TRUE.

Bucles.- Permiten realizar repeticiones de código o de condiciones de programa. En PHP se pueden realizar bucles while, for, y combinaciones do-while, y for-each. La sintaxis de un bucle while es la siguiente:

```
<php?

While (condición)

{
```

```
//acción que se repetirá hasta cumplir la condición
}
?>
```

Funciones.- Para declarar una función solo es necesario darle un nombre seguido de un paréntesis, como se muestra a continuación:

```
<php?
Función1()
{
//acciones de la función
}
?>
```

Con estos elementos básicos de programación en PHP, podemos realizar el script necesario para el almacenamiento de los votos de la aplicación NCL-Lua. Se debe tener en cuenta que PHP corre en un servidor por lo cual se debe incorporar este servicio a un computador, ya sea en Linux o en Windows.

4.2 Plataforma de contenido digital y gestión de usuarios

La plataforma de contenido que debe gestionarse, debe ser capaz de administrar las conexiones TCP entre el usuario y el servidor, así también como la interactividad en el televisor. Para seleccionar adecuadamente el servidor requerido, debemos tomar en cuenta que debe cumplir con las siguientes características:

- Manejo de conexiones TCP
- Servidor de http
- Servidor de base de datos
- Servidor de PHP

El paquete de software que incluye todas estas características y que es de código abierto es XAMPP de la organización ApacheFriends.

Con el servidor XAMPP en conjunto con las herramientas utilizadas en el capítulo anterior como VLC Media Player, OpenCaster, y el STB Ginga se diseñó una solución de red para la transmisión de un canal de televisión digital con retorno. La

estructura de red diseñada para cumplir con este propósito es la que se muestra en la figura.

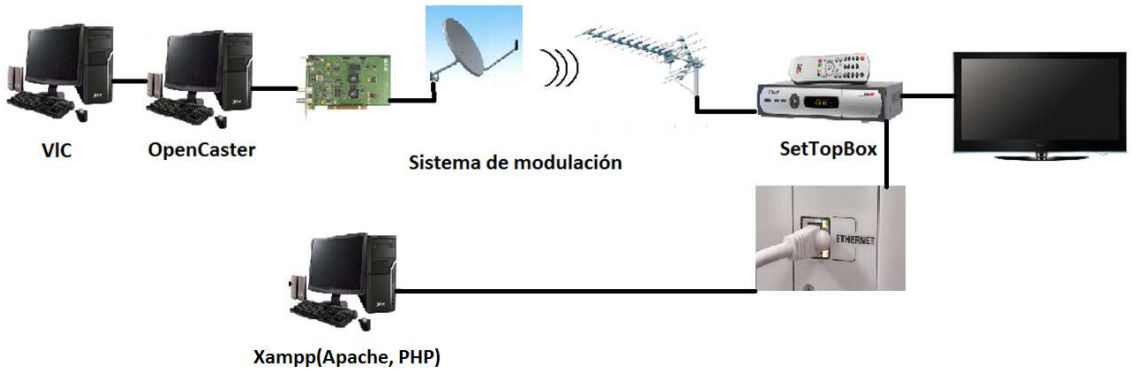


Figura 4.2. Esquema básico para la transmisión de un canal de televisión digital con retorno utilizando software libre

4.2.1 Servidor XAMPP

XAMPP es una plataforma web "cruzada" como una distribución del servidor http Apache, que permite su interconexión con MySQL, PHP y Perl. XAMPP ofrece al administrador del servidor una manera muy fácil y con interfaz gráfica para la el manejo de archivos, gestión de conexiones y activación y desactivación de servicios.

Figura 4.3. Logotipo de XAMPP

XAMPP utiliza Apache como su servidor http. Apache es una distribución libre de Apache Software Fundation creado en 1995, que permite la administración de múltiples páginas web en un mismo servidor, además permite una redirección automática para los dominios utilizados, y el control de puertos para acceso. Apache es uno de los servidores web más utilizados en el mundo debido a su facilidad de instalación, puesta en marcha y mantenimiento.

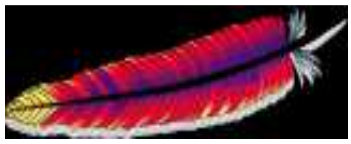


Figura 4.4. Logotipo de Apache

Por otro lado MySQL (My Structured Query Language) es una distribución de software libre de MySQL AB de Oracle creado en 1995, que permite la administración de múltiples bases de datos, y el acceso de múltiples usuarios a las mismas.

Figura 4.5. Logotipo de MySql

4.2.1.1 Instalación de XAMPP, Apache y MySQL [26]

La instalación de XAMPP en Windows o en Linux se hace de manera directa, descargándose los archivos de instalación de http://www.apachefriends.org/en/xampp.html. En Windows, luego de realizar la descarga, se procede a ejecutar el instalador, y seleccionar el lenguaje de instalación.

Figura 4.6. Lenguaje de instalación

El asistente nos guiará en todos los pasos, por lo cual hacemos click en siguiente en las ventanas de configuración.

Figura 4.7. Proceso de instalación

Es muy importante que la instalación se la haga en un nivel elevado dentro del disco duro, es decir no se debe instalar XAMPP dentro de subcarpetas, pues Apache no podrá ejecutar las páginas web guardadas.

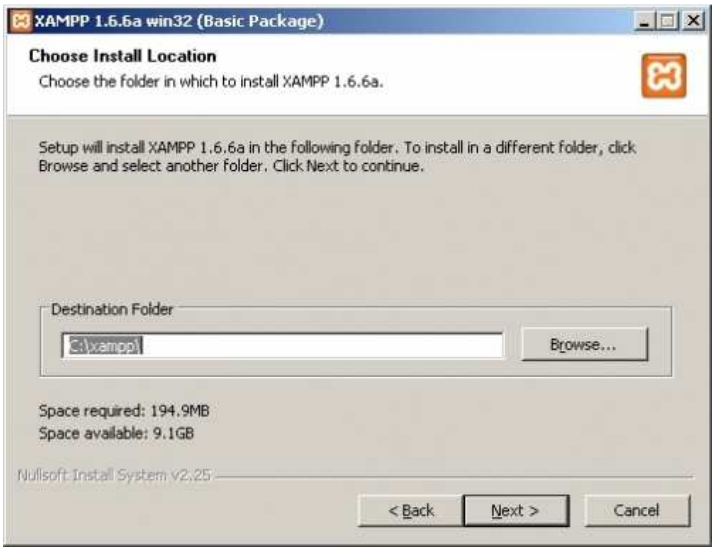


Figura 4.8. Carpeta de destino de XAMPP dentro del disco duro

Luego de indicar el directorio de instalación, se seleccionan los servicios que requerimos, para este caso debemos instalar Apache y MySQL. Cabe recalcar que este procedimiento hace que XAMPP inicie automáticamente al encender el servidor, por lo cual demanda que el mismo sea de características de memoria elevadas. Si requerimos que el servicio de Apache sólo se ejecute cuando se requiera y no al inicio de Windows, no se deben marcar las casillas que se muestran en la figura 4.9.

Figura 4.9. Instalación de Apache y MySQL como servicio predeterminado

Luego de la instalación podemos acceder al Panel de Control de XAMPP que se muestra en la figura. Como se puede observar, para activar o desactivar un servicio, es necesario hacer click en los botones de “Start” o “Stop”.

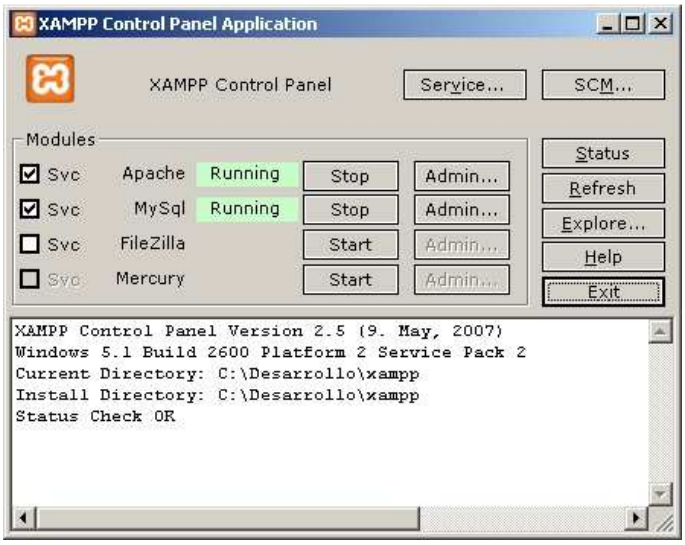


Figura 4.10. Panel de Control de XAMPP

Una característica importante de XAMPP es que se puede acceder al panel de control mediante HTTP desde otro computador remoto, para lo cual ingresamos en el buscador la dirección IP del servidor, o si estamos en el mismo computador se ingresa mediante http://localhost . La figura 4.11 muestra la interfaz HTTP que nos indica la correcta instalación de XAMPP, y además nos permite administrar los módulos instalados.

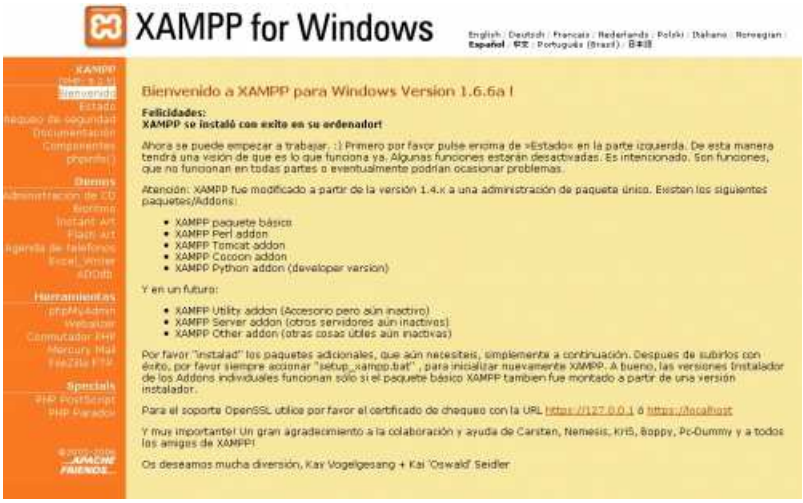


Figura 4.11. Interfaz HTTP de XAMPP

4.2.2 Diseño NCM con retorno

De igual manera que en el capítulo anterior, se debe diseñar la aplicación partiendo del diagrama en NCM. El documento NCM diseñado se muestra en el anexo.

En comparación con el diseño NCM de un canal de televisión digital abierto, en un canal con retorno, obligatoriamente debemos utilizar scripts en Lua para manejar las conexiones TCP. En este caso, los identificadores de la aplicación Lua interactúan con los puertos de NCL.

Un Script Lua es un elemento de la clase application/x-Ginga-NCLua que se decodificará dentro del documento NCL, y se ejecutará por el núcleo común de Ginga, específicamente por el Parser. La diferencia de este tipo de elemento multimedia es que puede ser ejecutado mediante valores limitados a comparación de otros elementos multimedia como imágenes y videos. Los valores que se pueden cargar en los links de función de un objeto multimedia Lua, son los que se muestran en la tabla.

Tabla 4.3. Valores de roles para elementos multimedia tipo x-Ginga-NCLua [28]

Valor	Acción	Tipo de evento
start	Inicio	presentación/ejecución
stop	Paro	presentación/ejecución
abort	Paro abrupto	presentación/ejecución
pause	Pausa	presentación/ejecución
set	Inicio	Cargar atributos

Específicamente se utilizará el valor “set” para cargar las votaciones en el archivo Lua. Cabe recalcar que esta aplicación Lua, sólo se ejecutará una vez y luego deberá ser actualizada por el carrousel de objetos. El televidente observará las opciones, y luego de realizar su votación, podrá observar los resultados parciales hasta el momento de realizada la misma.

4.2.3 Diseño NCL Lua [29]

Una vez diseñado el diagrama NCM para la aplicación, debemos trasladarlo a lenguaje NCL, y al Script en Lua. Para trasladar el diagrama NCM a NCL se utiliza la herramienta Composer como se mencionó anteriormente. En el capítulo 3.3.2 se especificó la manera de crear un documento NCL a partir del diagrama NCM, por lo que siguiendo este procedimiento se llegó a obtener el diagrama NCL mostrado en la figura 4.12.

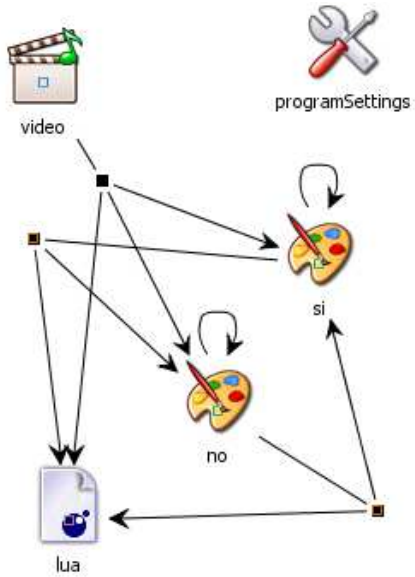


Figura 4.12. Diagrama NCL de la aplicación con canal de retorno

Como podemos observar, la aplicación NCL es relativamente simple, pues todo el control de interacción e interconexión se lo realiza en el script en Lua.

Para crear un elemento multimedia de tipo x-Ginga-NCLua, simplemente agregamos el nodo multimedia, pero seleccionamos “application/x-Ginga-NCLua” en la opción “Media Type” en el menú contextual de Composer.

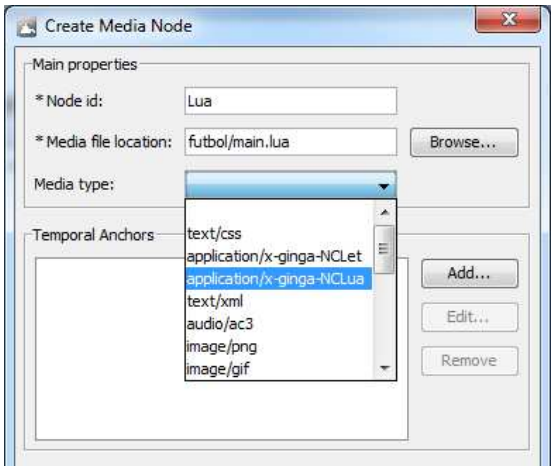


Figura 4.13. Creación de un nodo multimedia NCLua

Al momento de crear los Links causales entre los nodos multimedia comunes y un nodo Lua, debemos seleccionar la opción “OnBeginSetN” o “OnStopSetN”, para que las opciones dentro del documento Lua, puedan obtener los datos del voto realizado, o en general, del elemento multimedia seleccionado.

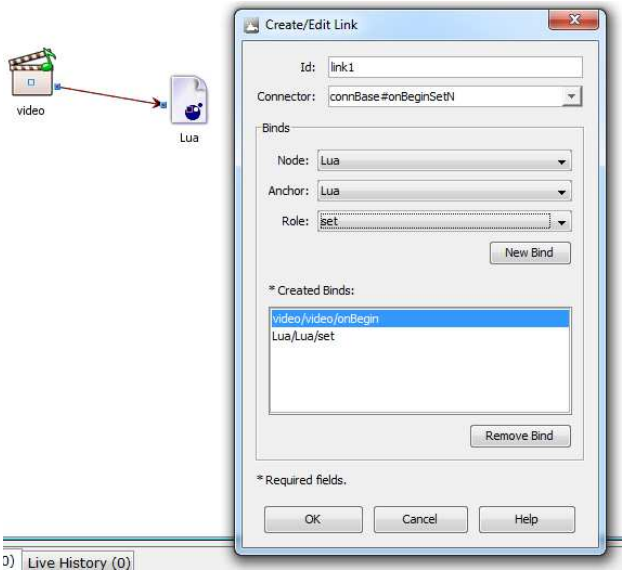


Figura 4.14. Link causal entre un elemento multimedia de tipo video y un script Lua

En la figura 4.15 se puede observar la pantalla principal donde se realiza la encuesta al televidente. Esta aplicación se ejecutará una sola vez, luego de que este proceso culmine se puede enviar al carrousel otra aplicación.

Figura 4.15. Pantalla principal de la aplicación de la encuesta

Luego de que se realice el voto, la aplicación se conecta por TCP al servidor PHP, y mostrará en pantalla los resultados parciales, como se muestra en la figura 4.16.

Figura 4.16. Respuesta TCP al usuario

Capítulo V:

SIMULACIÓN Y PRUEBAS DE FUNCIONAMIENTO

Introducción

El proceso de transmisión de un canal de televisión digital utilizando la norma ISDB-t debe cumplir con los parámetros descritos en los capítulos anteriores. Debido a que en el presente trabajo de tesis no se incorpora la plataforma física ni el hardware necesario para una transmisión de este tipo, se realizaron simulaciones y pruebas de funcionamiento sobre los flujos de transporte, la ejecución de la aplicación, y los tamaños y tasas de bit de los elementos de audio y video utilizados.

Uno de los parámetros más importantes a analizar es el tamaño de la trama enviada dentro del flujo de transporte, debido a que no se puede exceder el tamaño establecido en la norma para la transmisión, pues se ocasionaría que el equipo receptor no pueda cargar toda la información dentro del carrousel de objetos. Para este análisis se utilizará un paquete de software libre llamado “SBTVD Parser” el cuál se describirá en detalle en los siguientes subcapítulos.

Para el análisis del flujo de transporte sobre la capa IP se utilizará el paquete de software gratuito “WireShark” con el cual se puede interpretar las tramas de red involucradas, así también como el direccionamiento y las peticiones de la clase TCP Lua.

Por último, para la simulación de las aplicaciones, se utilizaron dos programas ya mencionados anteriormente: 1) la máquina virtual de Ginga en Kubuntu del LIFIA y 2) la máquina virtual de Ginga en Ubuntu Server de Telemidia. En estos ambientes de simulación se asegura una compatibilidad muy aceptable sobre plataformas físicas, por lo que se puede validar el funcionamiento correcto de las aplicaciones.

5.1 Ejecución de la aplicación de canal abierto

Debido a que las aplicaciones de canal abierto no envían datos a través de internet, la simulación de las mismas se realizó únicamente dentro de las máquinas virtuales, es decir, son sólo pruebas de funcionamiento de los algoritmos, y de la diagramación NCM. Se diseñaron dos tipos de aplicaciones para canal abierto, con la finalidad de mostrar las cualidades de la programación en NCL Lua. La primera aplicación se basa únicamente en imágenes que forman los menús de navegación, y la segunda

aplicación además de imágenes incorpora textos que pueden ser modificados por el servidor (a manera de teletexto) durante la transmisión.

La pantalla principal de la aplicación de imágenes, denominada “Futbol1”, se muestra en la figura 5.1. En esta pantalla el usuario sólo observa el botón de información, lo cual permite que se tenga una visión amplia del video.

Figura 5.1. Pantalla principal de la aplicación Futbol1

Cuando el usuario presiona el botón “Enter”, o el botón de “Info” en el control remoto del televisor, el formateador se encarga de minimizar la región de video a una más pequeña, y mostrar los index 1 y 2 dentro de la programación NCL. La pantalla resultante es la que se muestra en la figura 5.2.

Figura 5.2. Pantalla del menú

En este punto se inician los conectores de interactividad, permitiendo que Ginga interprete qué botón es presionado. En el caso de presionar el botón rojo, se mostrará la formación del equipo local. Cabe recalcar que estas imágenes deben ser previamente editadas, lo cual no es el caso si se manejan únicamente textos. En la figura 5.3 se muestra la pantalla correspondiente al botón rojo.

Figura 5.3. Menú rojo

Si se presiona el botón verde, se mostrará la formación del equipo visitante, de igual manera, esta imagen es previamente editada. En la figura 5.4 se muestra la pantalla correspondiente.

Figura 5.4. Menú verde

En el caso de presionar el botón amarillo, se mostrará la imagen correspondiente a los partidos de la fecha. En la figura 5.5 se muestra dicha pantalla.

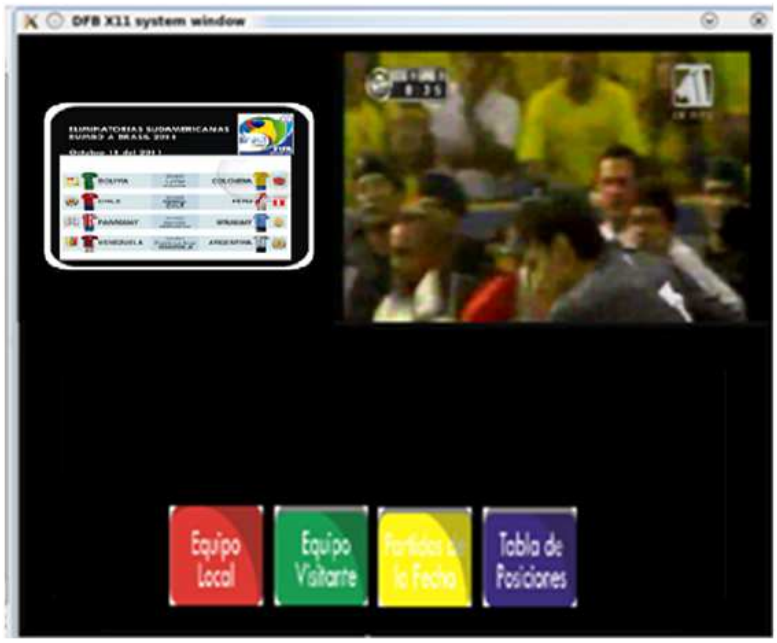


Figura 5.5. Menú amarillo

Por último, si se presiona el botón azul en el control remoto, el televidente observará la tabla de posiciones del campeonato. La figura 5.6 muestra la pantalla correspondiente al menú azul.

Figura 5.6. Menú azul

Todos los menús de esta aplicación son intercambiables, y se pueden activar en cualquier momento. Para hacer un cambio en las imágenes se debe realizar el menú por completo, pues no posee elementos textuales de edición.

5.2 Ejecución de la aplicación con canal de retorno

La aplicación utilizando el canal de retorno por Ethernet, como se mencionó en capítulos anteriores, se basa en una solicitud TCP entre el usuario y el servidor, en la cual se envía una encuesta básica sobre un aspecto específico en la transmisión. La pantalla principal (la que observa el usuario final) es la de la figura 5.13.

Figura 5.13. Pantalla principal de la aplicación DeportesRetorno

Como se mencionó anteriormente, tanto el texto como las imágenes de “SI” y “NO” se pueden cambiar para cualquier tipo de encuesta relacionada. La ejecución correcta de las tramas TCP se realizó con el paquete de software WireShark (figura 5.14).

Figura 5.14. Logotipo WireShark

En la ejecución de la clase TCP dentro del archivo Lua, existen dos fases principales: el envío de datos (voto realizado) el cual es una petición TCP realizada por el equipo usuario; y la solicitud de los votos totales, la cual se hace con la opción TCPGet dentro de la librería TCP.

Luego de realizar la petición de una conexión TCP con el servidor, el equipo usuario realiza el envío de datos. Este paquete debe tener una longitud de 74 bytes, con su respectivo encabezado de red, encabezado TCP, y por lo tanto las direcciones IP de los equipos (incluido el número de puerto), todos estos parámetros se los configura en la clase TCPLua, la cual se encarga de anexar los datos del voto realizado en la trama TCP. En la figura 5.15 podemos observar los datos capturados de una de las tramas TCP de envío de datos de la aplicación.

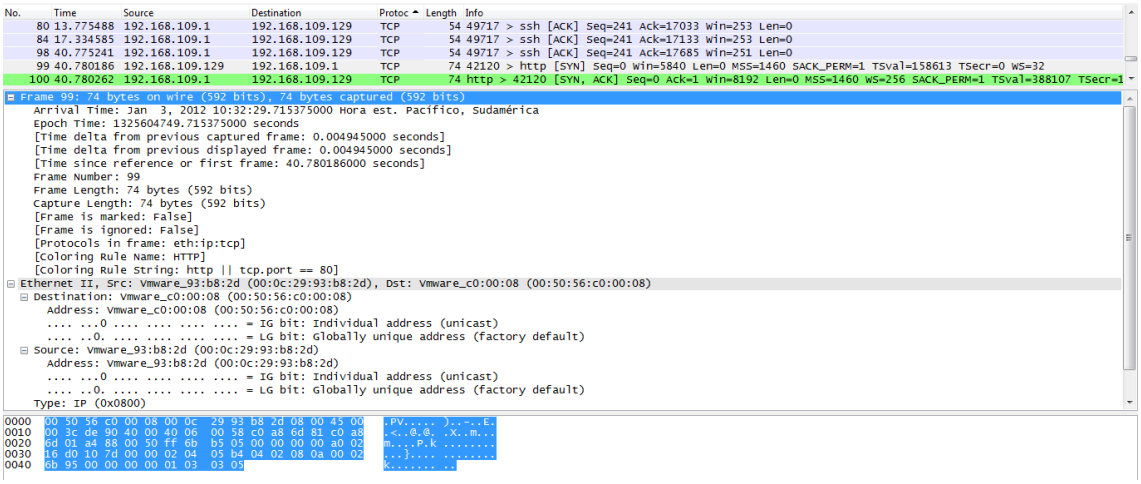


Figura 5.15. Trama TCP de envío de votos

El encabezado IP, en el cual se encuentran las direcciones de destino y fuente, se muestra en la figura 5.16.

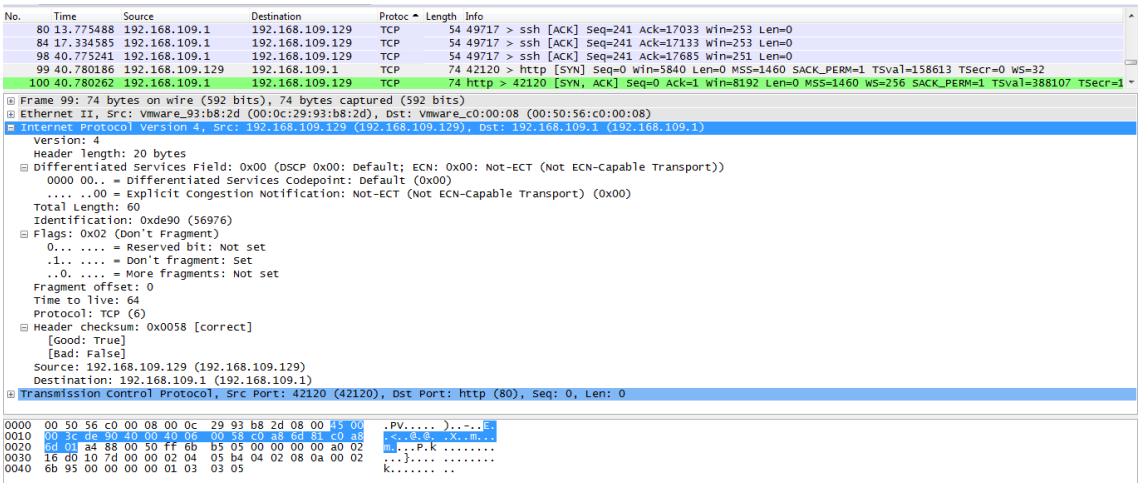


Figura 5.16. Trama IP de envío de votos

Una vez notificado el voto al servidor, la aplicación se encarga de mostrar en pantalla los resultados parciales hasta el momento en el que el usuario realizó su voto. Este proceso se realiza mediante la subclase TCPGet de la clase TCPLua, en el cual el equipo de usuario realiza otra petición TCP al servidor, pero esta vez para leer el archivo “No.txt” o “Si.txt” que se genera por el archivo PHP. El segmento TCP de solicitud es de 112 bytes de longitud, por lo cual se nota el retardo al momento de realizar la respuesta desde el servidor. En la figura 5.17 se muestran las características de la trama TCP de petición desde el equipo de usuario.

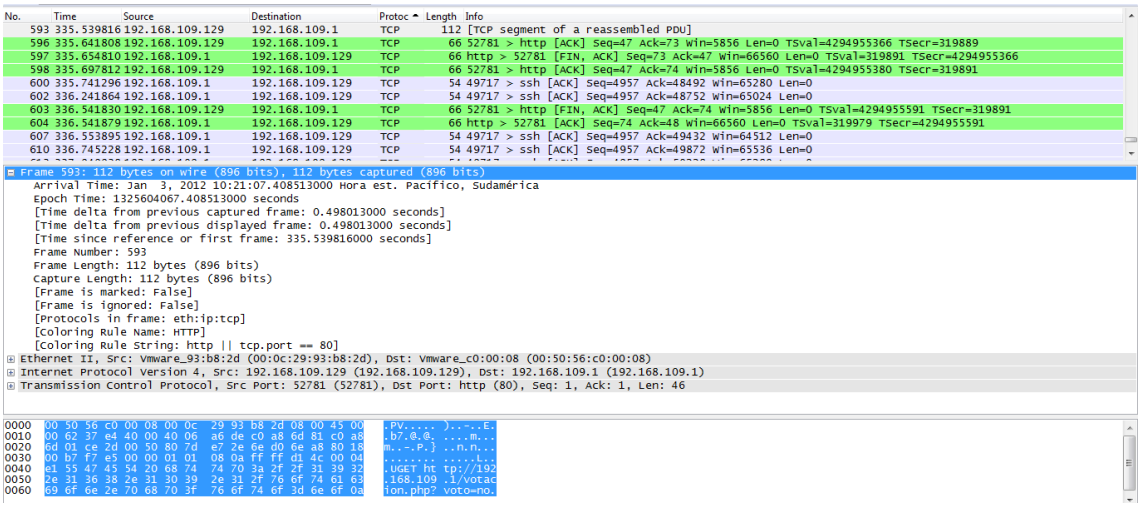


Figura 5.17. Trama TCP de solicitud de datos

Se puede observar el paquete “Get” de http que se envía al servidor que se encuentra, en este caso, en la dirección 192.168.109.1, y se puede observar el voto realizado, en este caso “voto=no”. En la figura 5.18 también podemos observar la estructura de la trama IP, en la cual se encuentran las direcciones de red utilizadas.

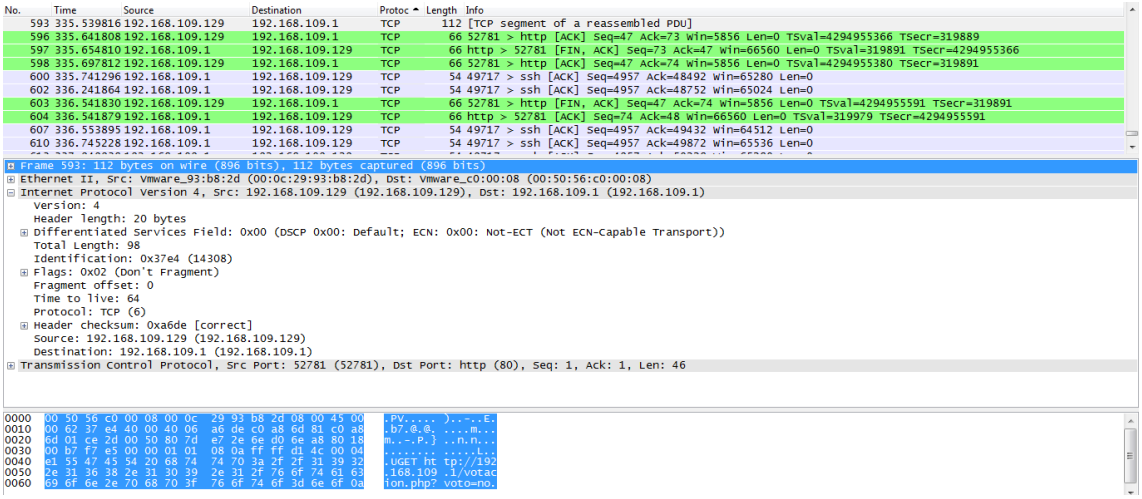


Figura 5.18. Trama IP de solicitud de datos

5.3 Análisis del flujo de transporte

Para asegurarnos que estamos cumpliendo con los parámetros de transmisión descritos en la norma ISDB-t, debemos analizar el "transport stream" o flujo de transporte generado. Para analizar este archivo nos valemos de una herramienta gratuita llamada "SBTVD Parser" (Figura 5.19).

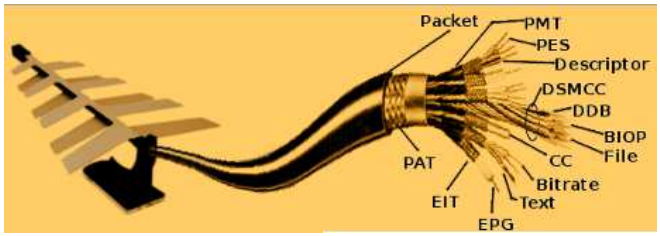


Figura 5.19. Logotipo de SBTVD Parser [32]

Características de SBTVD Parser [32]

SBTVD Parser es una herramienta gratuita que permite analizar el flujo de transporte de un sistema de televisión digital basado en la norma ISDB-tb, el cual permite:

- Detalle de la estructura de tablas PSI/SI de manera jerárquica
- Estadísticas de Bitrate para cada ES
- Decodificación de características de carrousel de objetos e IIP.
- Detección de eventos de EPG
- Decodificación de características DSMCC de carrousel

Estas características permiten analizar la gran mayoría de aspectos básicos en la transmisión de un flujo de transporte para televisión digital brasileña. El flujo de transporte utilizado para este propósito se generó utilizando OpenCaster como se especificó en capítulos anteriores y se procedió a su análisis.

El primer análisis requerido para asegurar la correcta transmisión del flujo de trasporte al carrusel de objetos es observar que las tablas PSI/SI estén correctamente cargadas.

Características claves de transmisión [34]:

- Tanto las tablas PAT como la PMT deber ser transmitidas por lo menos 10 veces por segundo, cada una de las tablas entra en un solo paquete de 188 bytes, debido a que se debe enviar para este caso 8 tramas, lo que da un total de 15040bps.
- El ancho de banda de ISDBT es de 29958294bps en paquetes de 188 bytes, o de 32507937 bps en paquetes de 204 bytes.

Teniendo esto en cuenta podemos analizar la tabla de programación PAT, los valores que se deben comprobar son los siguientes:

Valores Cargados por el proveedor de servicios:

- Número de versión=0
- Número de sección=0

Valores propios de la norma [33]:

- Pid =0x00
- Table_id=0x00

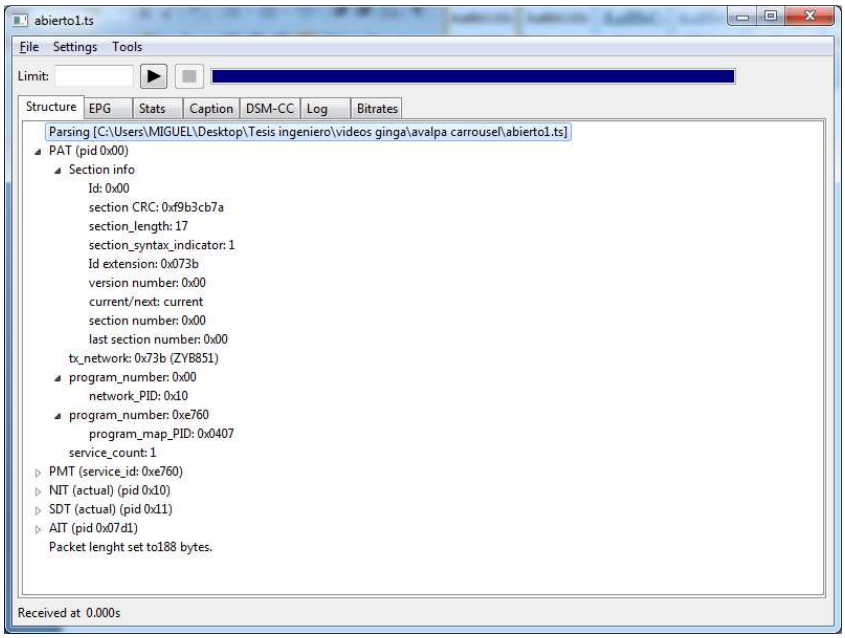


Figura 5.20. Valores de la tabla PAT en el flujo de transporte analizado

Las tablas de asociación de programa PMT (Program Map Table) deben tener los siguientes valores:

Valores Cargados por el proveedor de servicios:

- PID=0xe760
- PCR_PID=0x080f
- Version=0
- Sección=0

Valores propios de la norma [33]:

- Tableid=0x02
- Stream type=3 (audio-mpeg2)
- Stream type=2 (Video- mpeg2)

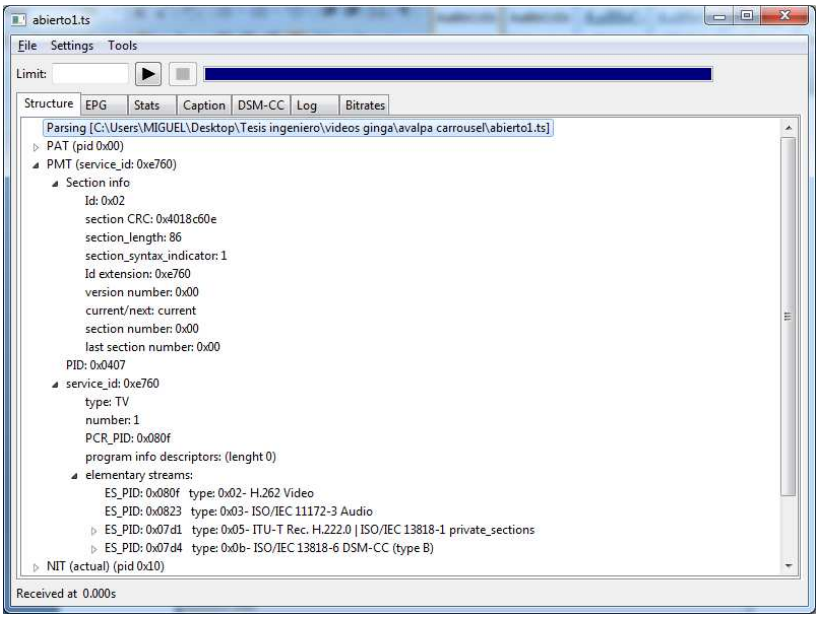


Figura 5.21. Valores de la tabla PMT en el flujo de transporte analizado

El encabezado de la tabla de red NIT (Network Information Table) según lo especificado en el capítulo III, debe poseer los siguientes valores:

Valores Cargados por el proveedor de servicios:

- Network id=0x073b
- Network descriptor= “LIFIATV”
- Transport stream id= 0x073b
- Service id=0xe660
- Service type=1

Valores propios de la norma [33]:

- PID=0x0010
- Table_id=0x40

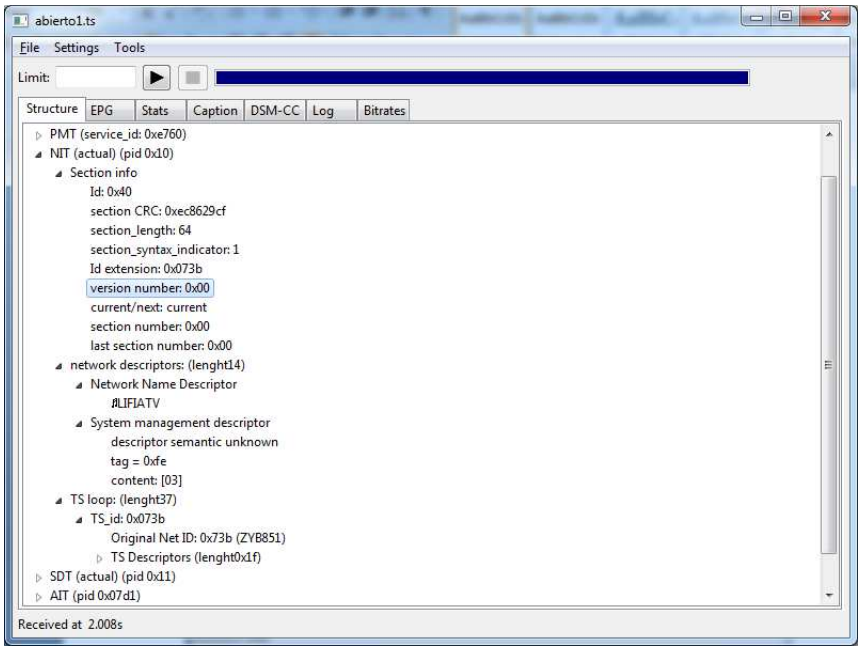


Figura 5.22. Valores de la tabla NIT en el flujo de transporte analizado

La tabla de descripción de servicios SDT (Service Description Table), debe poseer los valores de los identificadores de todos los servicios dentro del flujo de transporte, los cuales son:

Valores Cargados por el proveedor de servicios:

- EIT Schedule flag=0
- EIT present following flag=0
- Service provider name= “”

Valores propios de la norma [33]:

- Pid=0x0011
- Table_id=0x42
- Service type=2

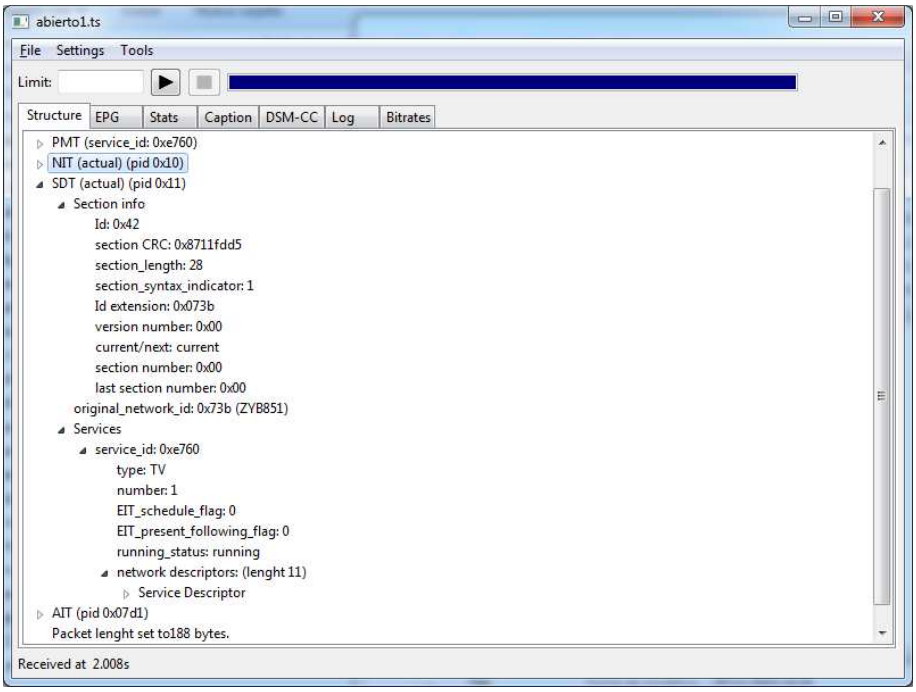


Figura 5.23. Valores de la tabla SDT en el flujo de transporte analizado

Por último, la tabla de aplicación AIT (Aplication Information Table) debe tener la información de la aplicación interactiva que se transmite, los parámetros más importantes son:

Valores Cargados por el proveedor de servicios:

- Application_type=0x0009 (Ginga ncl)
- Organization id=0x0000000A
- Application id=0x74
- Application control code 0x01

Valores propios de la norma [33]:

- Protocol id=0x0001
- PID=0x07d1

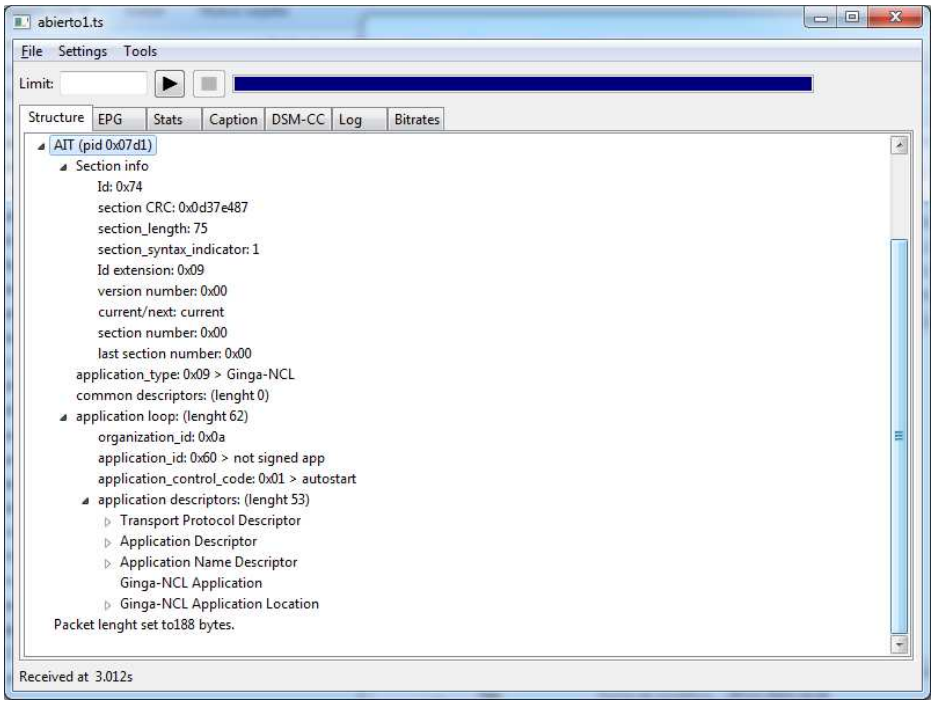


Figura 5.24. Valores de la tabla AIT en el flujo de transporte analizado

Otra característica que se puede analizar con este software es la guía de programación. En la presente aplicación no se requería la utilización de esta alternativa, pero el programa detecta que está configurado pero que no se tiene activo.

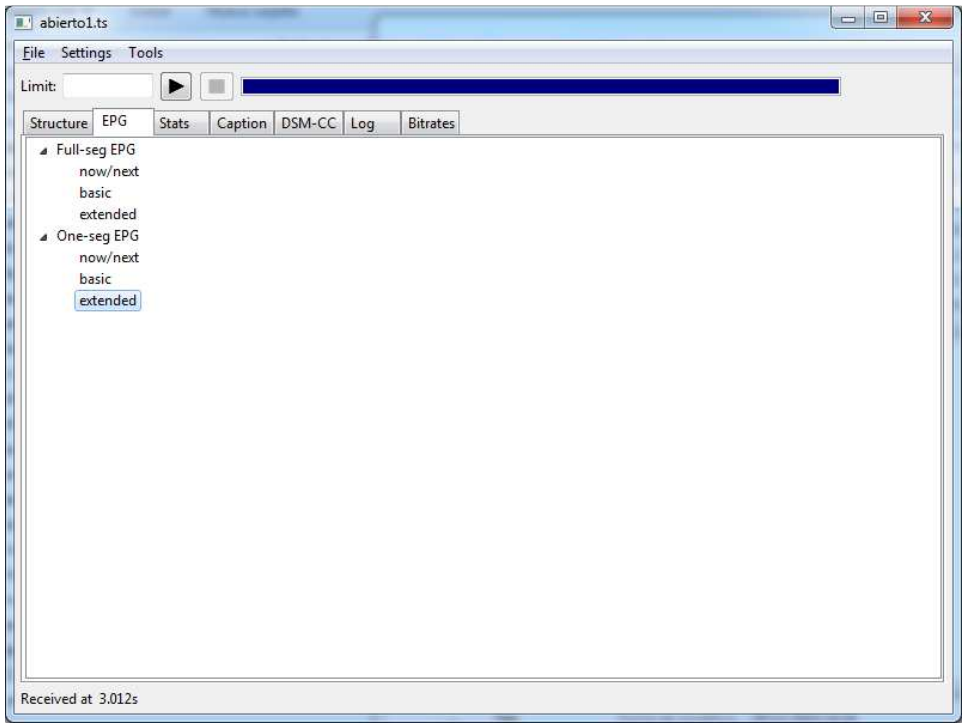


Figura 5.25. Análisis de la guía de programación

Tasa de bits

Se analizan las tasas de bit por separado de cada tipo de tabla en la estructura. Los más importantes son los siguientes:

- Id=0x08f, correspondiente a video h262
- Id=0x07d4, correspondiente a la actualización del carrousel de objetos dsm-cc tipo b
- Id=0x1fff, correspondiente a la sincronización por pcr pid

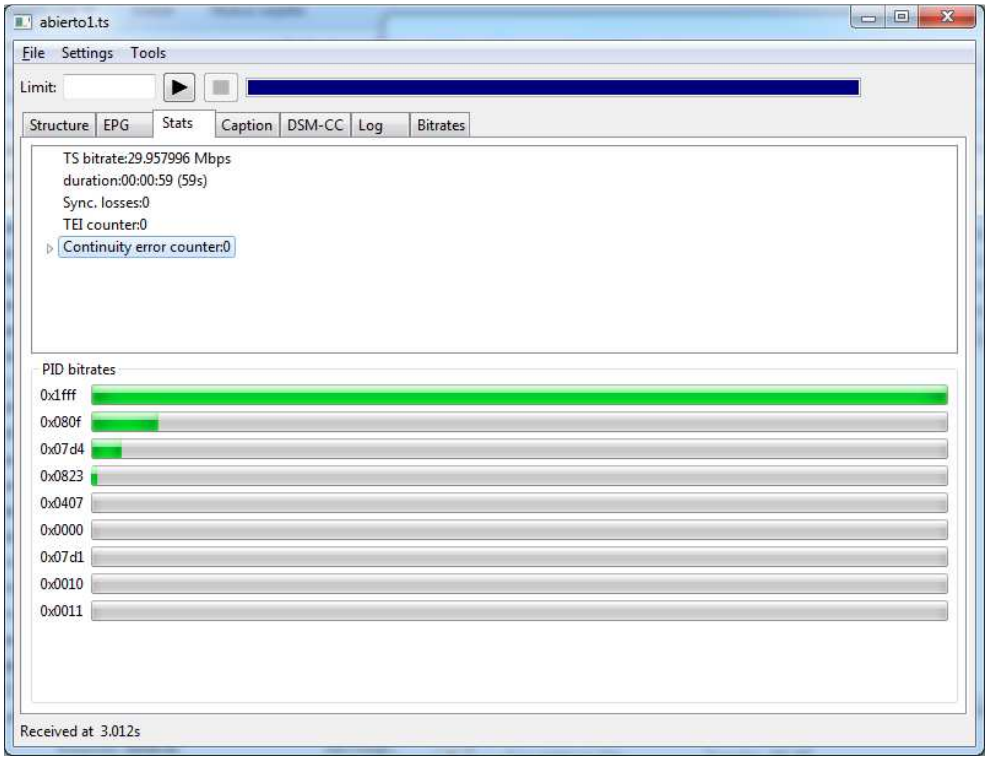


Figura 5.26. Representación de las tasas de bits correspondientes a los identificadores más utilizados en el programa.

Analizando la figura 5.26 podemos darnos cuenta que el programa realiza un uso eficiente de ancho de banda, apenas alrededor del 25% del total, esto se debe a que las imágenes son livianas (en formato png) y el audio está codificado. Este resultado es el esperado, pues estamos transmitiendo un canal de resolución estándar, y dos servicios dentro del flujo de transporte. Si se requiere transmitir un canal de resolución mayor, se debe tener en cuenta que los bitrates se elevan sustancialmente debido no sólo a la codificación del audio y el video, sino también a las imágenes utilizadas e incluso el manejo de scripts para las animaciones.

También podemos hacer el análisis específico del contenido del carrousel de objetos. En la figura 5.27 se muestra el análisis principal del contenido del carrousel para la aplicación diseñada.

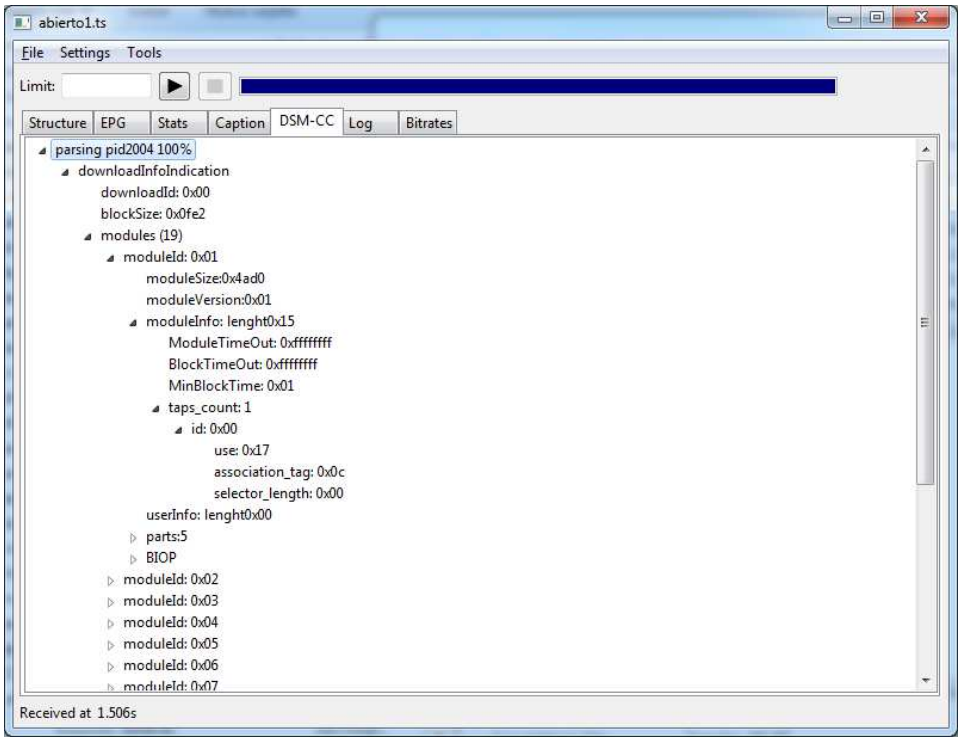


Figura 5.27. Análisis del carrousel de objetos

Este análisis permite revisar cada módulo del carrousel de objetos, lo cual es una herramienta poderosa para encontrar errores en caso de que una aplicación no se cargue correctamente en un equipo de usuario. En la figura 5.28 podemos observar el contenido de las aplicaciones ncl, y las imágenes utilizadas en la misma, lo cual se cargará en el carrousel de objetos.

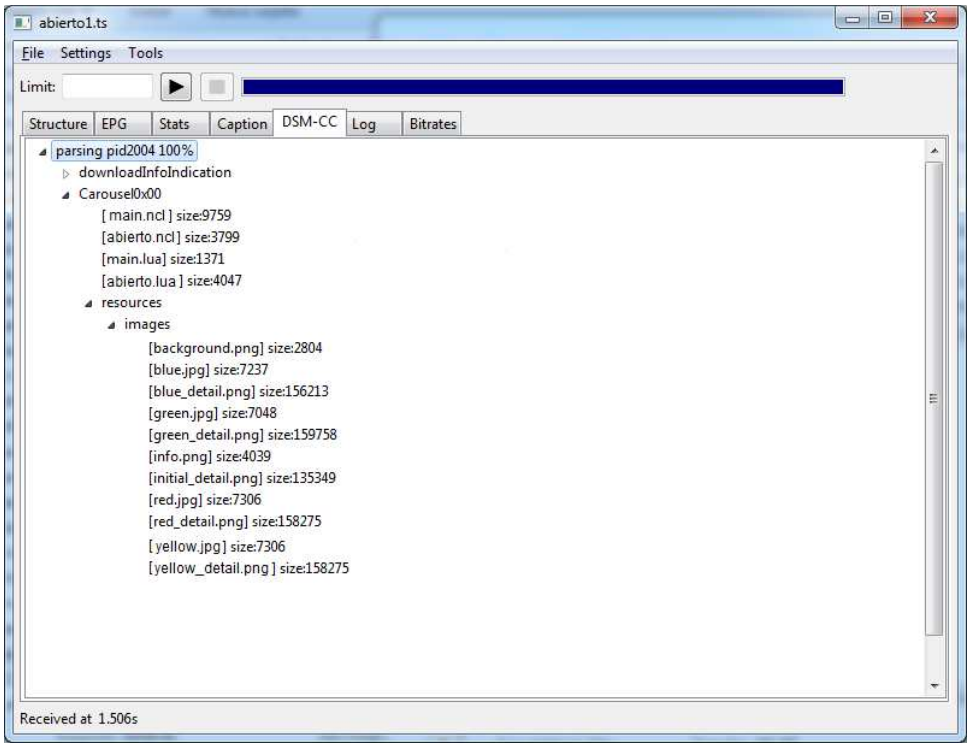


Figura 5.28. Especificaciones de la aplicación en el contenido del carrousel de objetos

Por último, se puede analizar los PIDs de la aplicación en una gráfica que nos muestra la frecuencia de actualización y el ancho de banda utilizado. Se analizaron los componentes de audio, video y del carrousel de objetos.

El espectro de video se muestra en la figura 5.29, donde se puede medir un máximo de 2.75Mbps y una tasa constante de 2.12Mbps consumidos por el canal principal. Este análisis es importante, pues se debe mantener una tasa de bits constante para la transmisión.

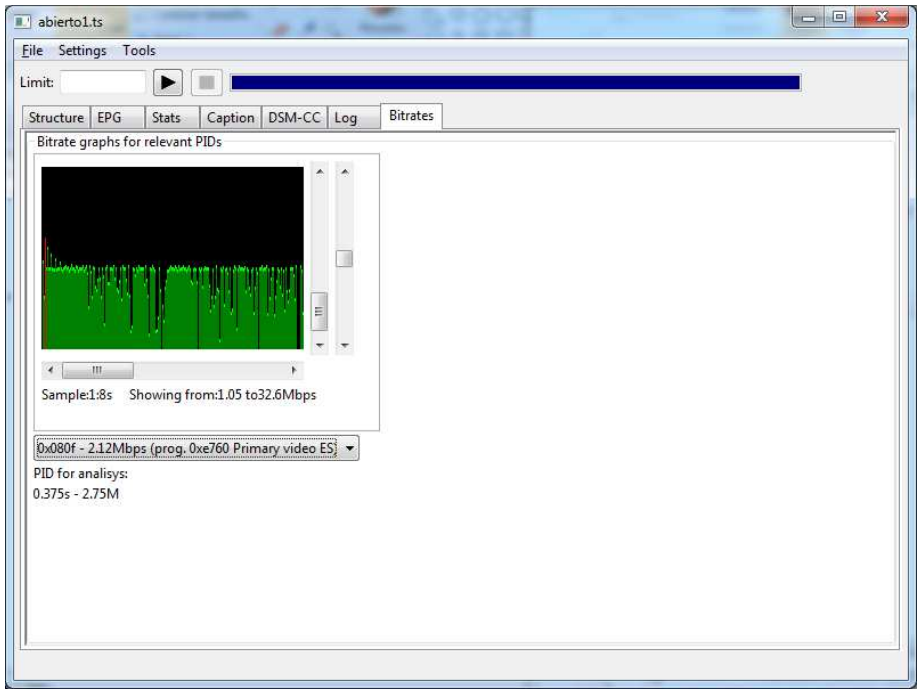


Figura 5.29. Tasa de bits del video en el carrousel

En el caso de la actualización del carrousel de objetos, se tiene un espectro máximo de 1.2Mbps, y una tasa constante de 1Mbps, lo cual se encuentra dentro del rango establecido por la norma. Este resultado se puede observar en la figura 5.30.

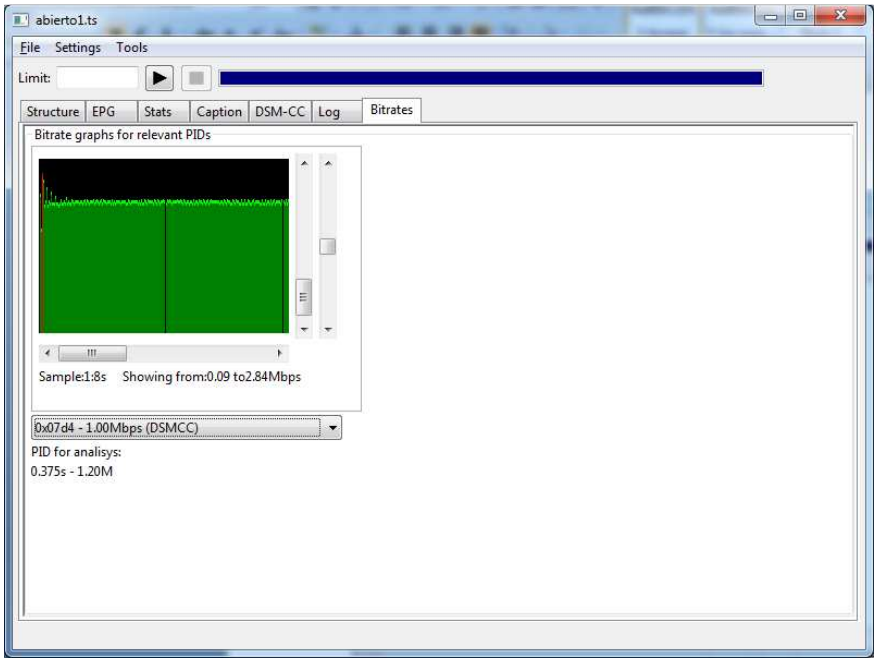


Figura 5.30. Tasa de bits de la actualización del carrousel de objetos

Por último, para el audio, se tiene una tasa de 188kbps, lo cual está dentro del estándar MPEG4 como dicta la norma brasileña. Esto se puede observar en la figura 5.31.

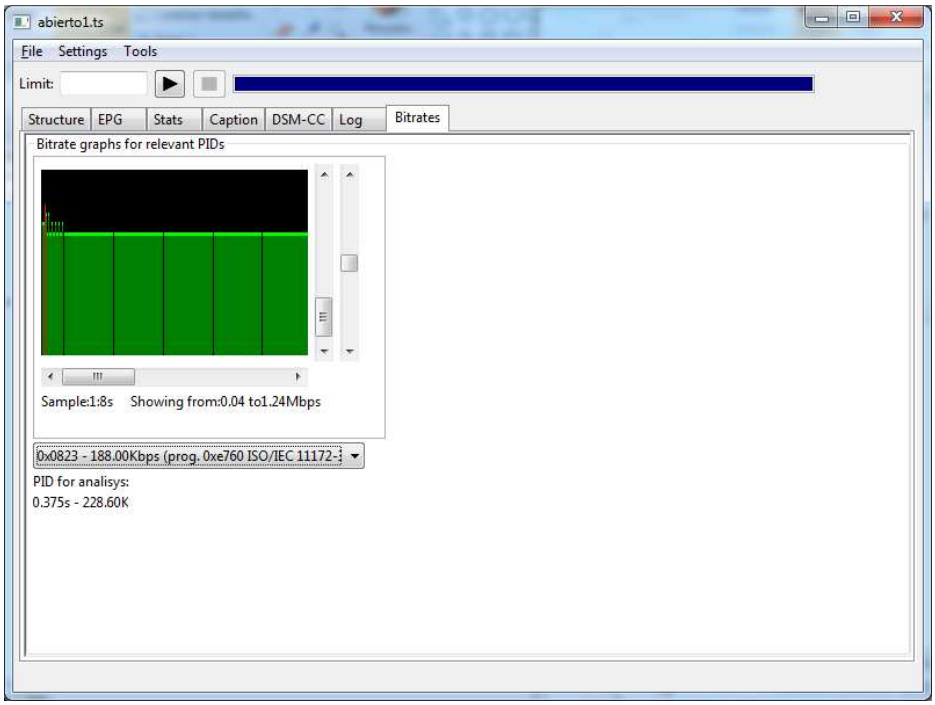


Figura 5.31. Tasa de bits del audio en el carrousel

Estos análisis son los más básicos para poder enviar una transmisión al aire sin problemas, pero se debe tener en cuenta que cuando trabajamos con más canales a la vez, o con aplicaciones más pesadas se deben ajustar los anchos de banda ocupados por cada PID, logrando que el carrousel pueda actualizarse en los tiempos debidos, caso contrario la aplicación no se ejecutará en el equipo receptor.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones Generales:

- El diseño de aplicaciones interactivas para televisión digital es un campo tecnológico que demanda muchos conocimientos de programación, diseño y normativa, por lo cual el profesional a cargo de la gestión, y mantenimiento de este tipo de sistemas debe ser acorde con las exigencias mínimas planteadas en este trabajo de grado.
- En cuanto a la normativa, se deben respetar valores como tasas de bit, tamaño de trama, y bits de relleno pues una transmisión que altere estos valores no podrá ser cargada en los equipos terminales.
- El diseño de una aplicación utilizando NCL no representa una complejidad mayor a cualquier sistema computacional, o diseño de software en cuanto a algoritmos se refiere, lo que sí se debe tener en cuenta es la aplicabilidad, funcionalidad, facilidad de navegación, y estética de dicha aplicación, lo cual es el éxito de un programa de televisión.
- Para diseños mejor elaborados se debe tener una línea gráfica para la realización y diseño de menús interactivos, imágenes y botones, lo cual da uniformidad a las aplicaciones.

Conclusiones Técnicas:

Con motivo de que esta tesis es una de las primeras realizadas sobre televisión digital en la Universidad Politécnica Salesiana sede Cuenca, a continuación se realizarán algunas conclusiones sobre los programas utilizados, así se podrá dar continuidad al análisis y desarrollo de televisión digital.

- Se realizaron pruebas con tasas de bits variables en los videos, lo cual se puede ejecutar dentro de una máquina virtual, pero esto no es recomendable pues está fuera de la norma, y puede ocasionar errores de transmisión.
- No existe una manera de transmitir el carrousel de datos directamente a las máquinas virtuales proporcionadas por el Lifia o Telemidia, por lo cual se deben cargar las aplicaciones directamente por el protocolo SSH.
- Luego de varias pruebas en máquinas virtuales, se llegó a la conclusión que las de mejor desempeño son la proporcionada por el Lifia en su versión de Kubuntu, y la proporcionada por Telemidia en su versión en Ubuntu Server. Las otras máquinas virtuales probadas presentaban problemas en cuanto a la ejecución de ciertos comandos en Lua, o a las bibliotecas para abrir sockets de comunicación.
- También, luego de realizar varias pruebas con programas de autoría para televisión digital brasileña, se concluye que los mejores paquetes de software disponibles en el mercado son Composer de laboratorios Telemidia, y Berimbau iTV Author Beta de Batuque TV. Estos paquetes de software

permiten al usuario sin conocimiento en NCL o Lua realizar sus propias aplicaciones, además consiste un paquete básico en una cabecera de televisión digital, en conjunto con los programas de edición y autoría gráfica.

- Además del programa SBTVD Parser utilizado para analizar el flujo de transporte generado, no se pudo encontrar más programas para este propósito, por lo menos no programas gratuitos, por lo cual el diseño de herramientas de esta índole es vital para seguir en esta rama de investigación.
- La herramienta OpenCaster permite crear los flujos de transporte necesarios para la transmisión. Al ser una herramienta libre, se pueden realizar modificaciones en los drivers para las tarjetas moduladoras, por lo cual el uso del PXI de National Instruments dentro de la Universidad fuera una opción muy apropiada.
- Al utilizar VLC Media Player como un transmisor del flujo de video, se logra una configuración parecida a IPTV, por lo cual este programa no es el más recomendado para aplicaciones profesionales, pero si para pequeñas televisoras, o canales barriales debido que no tiene costo y es una herramienta de software libre.

Recomendaciones:

- El conocimiento previo de lenguajes de programación como Lua, NCL, PHP, HTML y Python debe ser un prerrequisito para el diseño de aplicaciones interactivas. Además es necesario manejar bases de datos con MySQL, servidores web como APACHE y programas de diseño gráfico como BLENDER, por lo que este campo multidisciplinario exige una vasta experiencia y un manejo muy completo de programación y creación de algoritmos.
- En la medida de lo posible, las pruebas deben realizarse en un set top box físico, pues las características de velocidad de datos en una máquina virtual son mayores lo cual nos da una perspectiva no muy real sobre el funcionamiento de una aplicación. Así también la ejecución de comandos de edición es completamente realizable en un dispositivo terminal físico, mientras esta característica no se puede ejecutar en una máquina virtual.

Referencias

[1] “Diseño y desarrollo de una aplicación de contenidos interactivos para TV Digital basada en el middleware Ginga del sistema brasileño”, TORRES ALTAMIRANO, JAVIER EDUARDO. Escuela Politécnica del Ejército, 2010. Sangolquí- Ecuador.

[2] “Investigación del Estudio del Middleware Ginga y Guía de usuario del Middleware Ginga”, INSTITURO NACIONAL DE INVESTIGACIÓN Y CAPACITACIÓN DE TELECOMUNICACIONES INICTEL-UNI. Universidad Nacional de Ingeniería, abril de 2010. Lima-Perú.

[3] “Sistema Brasileño de Televisión Digital Terrestre SBTVD-T”, FRANCO, OLIMPIO JOSÉ, 2010. Argentina.

[4] “Televisión Digital Terrestre e Interactividad”; PÉREZ, NELSON. Universidad de Los Andes. GITEL-ULA, noviembre de 2009. Caracas-Venezuela.

[5] “Estudio de factibilidad para la implementación de un laboratorio de televisión digital interactiva para la ESPE”, SAAVEDRA ABARCA, EVELYN VANESSA. Universidad Politécnica del Ejército, 2009. Sangolquí-Ecuador.

[6] “Diseño de la red para interactividad en televisión digital terrestre e IPTV en el campus ESPE Sangolquí”; MORALES FIGUEROA, AMPARITO ALEXANDRA. Universidad Politécnica del Ejército, 2010. Sangolquí-Ecuador.

[7] “Transmission System for ISDB-T”; TAKADA, MASAYUKI y SAITO, MASAFUMI. Proceedings of the IEEE, Vol.94 no.1, Enero de 2006. ISBN 0018-9219.

[8] “An Introduction to DTV and to Ginga-NCL”, Anónimo. Laboratorio Telemidia, 2006. Rio de Janeiro-Brasil.

[9] “TV digital interativa no Brasil se faz com Ginga. Fundamentos, Padro~es, Autoria Declarativa e Usabilidade”; JUNQUEIRA, SIMONE y GOMES,LUIZ. Departamento de Informática de la Pontificia Universidad Católica de Río de Janeiro, 2008. Río de Janeiro-Brasil.

[10] ABNT NBR 15601 (2007) – Associação Brasileira de Normas Técnicas, “Televisão digital terrestre- Sistema de transmissão” Sistema Brasileiro de TV Digital Terrestre, NBR 15601.

[11] ABNT NBR 15602-1 (2007) – Associação Brasileira de Normas Técnicas, “Televisão digital terrestre- Codificação de video, áudio e multiplexação” Sistema Brasileiro de TV Digital Terrestre, NBR 15602-1.

[12] ABNT NBR 15606-5 (2007) – Associação Brasileira de Normas Técnicas, “Televisão digital terrestre- Codificação de datos e especificações de transmissão para radiodifusão digital” Sistema Brasileiro de TV Digital Terrestre, NBR 15608-5.

[13] “Aplicaciones NCL”, varios autores, comunidad en línea de diseño NCL. Documento en línea disponible en www.clubencl.org.br. Enero, 2011.

[14] “Overview of ISDB-T”, anónimo. Documento en línea disponible en http://www.dibeg.org/overview/. Marzo 2011.

[15] “Desarrollo de aplicaciones de TVD interactiva con Ginga”; GOMES SOARES, LUIZ FERNANDO. Pontificia Universidad Católica de Río de Janeiro. Documento en línea disponible en http://www.ramiropol.com.ar/ginga-desarrollo-de-aplicaciones-para-tv-digital/. Julio, 2012.

[16] “Construindo Programas Audiovisuais Interativos Utilizando a NCL 3.0”; SOARES, CARLOS; GOMES, LUIZ; FERREIRA, ROGÉRIO; y JUNQUIERA, SIMONE. Pontificia Universidad Católica de Río de Janeiro. Laboratorios Telemidia, 05 de 2010. Río de Janeiro-Brasil.

[17] “Manual de referencia de Lua 5.1”. Documento en línea disponible en http://www.lua.org/manual/5.1/es/manual.html, Junio de 2011

[18] “Análisis del Canal de Retorno para la Televisión Digital Interactiva utilizando la Clase TCP-Lua”; PAUCAR, RONALD; MENDOZA, EDUARDO; DÍAZ, DANIEL; VILLANUEVA, JUAN. Universidad Nacional de Ingeniería INICTEL-UNI. Instituto Nacional de Investigación y Capacitación en Telecomunicaciones, mayo 2011. Lima-Perú.

[19] “Curso Básico de Redes de Cable” Anónimo. Documento en línea disponible en http://www.krconsult.com/cursos/redes2/redcab.htm. Junio, 2011.

[20] “Implementación de una solución para la gestión de un canal de televisión digital terrestre”; FERNÁNDEZ, FERNANDO; y ANSORENA, PEDRO. Departamento de Ingeniería Electrica y electrónica UPNA. Universidad Pública de Navarra, Octubre de 2009. Pamplona, España.

[21] “Adobe Premiere Pro: Especificaciones técnicas”, documento en línea disponible en http://www.adobe.com/es/products/premiere/tech-specs.html. Agosto, 2011.

[22] “VSNSharer: Introduction” documento en línea disponible en http://www.vsn-tv.com/es/programa/overview/4/vsnsharer.html. Agosto, 2011.

[23] “MuxXpert SDK”, documento en línea disponible en http://www.dektec.com/Products/SDK/MuxXpert/index.asp. Agosto, 2011.

[24] “What is Lives?”, documento en línea disponible en http://lives.sourceforge.net/, Agosto 2011.

[25] “OpenCaster para SATVD-T”, Laboratorio de Investigación y Formación en Informática Avanzada LIFIA. Universidad Nacional, 3 de mayo de 2001. Buenos Aires- Argentina.

[26] “Installing, configuring and developing with XAMPP”. DVORSKI, DALIVOR. Documento en línea disponible en http://dalibor.dvorski.net/. Ontario-Canada.

[27] “MySQL for the newbie and the experienced user”. DUBOIS, PAUL; y WIDENIUS MICHAEL. Julio, 2011.

[28] “Apache: Strategy in the New Economy”. ABRAMZON, EMILIANO; BLOOM, RYAN; MOHRMAN, AARON; y WILLIAMS, COLIN. Documento en línea disponible en www.apache.org , Diciembre del 2003.

[29] “NCLua SOAP: Acceso a Web Services em Aplicações de TVD”. CAMPOS, MANUEL. Documento en línea disponible en www.manoelcampos.com. Julio, 2011.

[30] “Desenvolvimento de Aplicações Declarativas para TV Digital no Middleware Ginga com Objetos Imperativos NCLua”. SANT’ANNA, FRANCISCO; SOARES, CARLOS; GERSON, ROBERTO; y JUNQUEIRA, SIMONE. Departamento de informática de la PUC-Río, Octubre de 2009. Fortaleza-Brasil

[31] “Nested Composite Nodes and Version Control in an Open Hypermedia System Revisited”. SOARES, LUIZ; RODRIGUEZ, NOEMI; y CASANOVA, MARCO. Departamento de informática de la PUC-Rio, Noviembre de 1996. Sao Paulo-Brasil.

[32] “O projeto Analizador SBTVD é uma pequena ferramenta para profissionais que trabalham com TV Digital.” Gabriel A. G. Marques. Documento en línea disponible en http://sbtvdparser.sourceforge.net/index_pt.htm

[33] “Televisión digital terrestre- Guía de operación Parte 3: Multiplexación y servicio de información (SI)”ABNTNBR15608-3. Segunda edición. 2009.

[34] “OpenCaster para SATVD-T”, Laboratorios de Investigación y Formación en Informática Avanzada-LIFIA. 3 de mayo de 2011

Anexos

CAPITULO III

ANEXO 1 Script en Python para la configuración en OpenCaster

```
nit = network_information_section(
        network_id = tvd_orig_network_id,
        network_descriptor_loop = [
                network_descriptor(network_name = "LIFIATV",),
                system_management_descriptor(
                        broadcasting_flag = 0,
                        broadcasting_identifier = 3,
                        additional_broadcasting_identification = 0x01,
                        additional_identification_bytes = [],
                )
        ],
        transport_stream_loop = [
                transport_stream_loop_item(
                        transport_stream_id = tvd_ts_id,
                                original_network_id = tvd_orig_network_id,
                                transport_descriptor_loop = [
                                        service_list_descriptor(
                                                dvb_service_descriptor_loop = [
                                                        service_descriptor_loop_item (
                                                                service_ID =
                                                        tvd_service_id_sd,
                                                                service_type = 1,
                                                        ),
                                                ],
                                        ),
```

```
                terrestrial_delivery_system_descriptor(
                        area_code = 1341,
                        guard_interval = 0x01,
                        transmission_mode = 0x02,
                        frequencies = [
                                tds_frequency_item( freq=ts_freq )
                        ],
                ),
                partial_reception_descriptor (
                        service_ids = []
                ),
                transport_stream_information_descriptor (
                        remote_control_key_id = ts_remote_control_key,
                        ts_name = "tvdeportes",
                        transmission_type_loop = [
                                transmission_type_loop_item(
                                        transmission_type_info = 0x0F,
                                        service_id_loop = [
                                        service_id_loop_item(
                                        service_id=tvd_service_id_sd
                                        ),
                                        ]
                                ),
                transmission_type_loop_item(
transmission_type_info = 0xAF,
service_id_loop = [],
),
```

```
],
)
],
),
],
version_number = 0,
section_number = 0,
last_section_number = 0,
)
```

Anexo 2: main.ncl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="DeportesAbierto" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
	<head>
		<importedDocumentBase>
			<importNCL alias="introNCL" documentURI="inicio.ncl"/>
		</importedDocumentBase>
		<ruleBase>
			<rule comparator="eq" id="DespliegueRule" value="Despliegue"
var="stage"/>
			<rule comparator="eq" id="menurule" value="menu" var="stage"/>
			<rule comparator="eq" id="interactivityRule" value="ON"
var="interactivity"/>
		</ruleBase>
		<regionBase>
			<region height="100%" id="bgReg" left="0%" top="0%"
width="100%" zIndex="0"/>
			<region height="100%" id="videoReg" left="0%" top="0%"
width="100%" zIndex="1"/>
			<region height="37" id="infoReg" left="590" top="430" width="37"
zIndex="2"/>
			<region height="205" id="MenuReg" left="0" top="275"
width="100%" zIndex="2"/>
			<region height="275" id="DespliegueReg" left="0" top="0"
width="260" zIndex="2"/>
		</regionBase>
		<descriptorBase>
			<descriptor id="backgroundDesc" region="bgReg"/>
```

```
                <descriptor id="videoDesc" region="videoReg"/>
                <descriptor id="infoDescriptor" region="infoReg"/>
                <descriptor id="menuDesc" region="menuReg"/>
                <descriptor id="despliegueDesc" region="despliegueReg"/>
            </descriptorBase>
            <connectorBase>
                <causalConnector id="onKeySelectionStopSetStart">
                    <connectorParam name="keyCode"/>
                    <connectorParam name="var"/>
                    <simpleCondition key="$keyCode" role="onSelection"/>
                    <compoundAction operator="seq">
                        <simpleAction max="unbounded" qualifier="seq"
role="stop"/>
                        <simpleAction max="unbounded" qualifier="seq"
role="set" value="$var"/>
                        <simpleAction max="unbounded" qualifier="seq"
role="start"/>
                    </compoundAction>
                </causalConnector>
                <causalConnector id="onBeginStopSetStart">
                    <connectorParam name="keyCode"/>
                    <connectorParam name="var"/>
                    <simpleCondition key="$keyCode" role="onBegin"/>
                    <compoundAction operator="seq">
                        <simpleAction max="unbounded" qualifier="seq"
role="stop"/>
                        <simpleAction max="unbounded" qualifier="seq"
role="set" value="$var"/>
                        <simpleAction max="unbounded" qualifier="seq"
role="start"/>
```

```
                    </compoundAction>
                </causalConnector>
                                    <causalConnector id="onKeySelectionSet">
                    <connectorParam name="keyCode"/>
                    <connectorParam name="varSet"/>
                    <simpleCondition key="$keyCode" role="onSelection"/>
                    <simpleAction role="set" value="$varSet"/>
                </causalConnector>
                <causalConnector id="onEndAttributionStopSetStart">
                    <connectorParam name="var"/>
                    <simpleCondition role="onEndAttribution"/>
                    <compoundAction operator="seq">
                        <simpleAction max="unbounded" qualifier="par"
role="stop"/>
                        <simpleAction role="set" value="$var"/>
                        <simpleAction max="unbounded" qualifier="par"
role="start"/>
                    </compoundAction>
                </causalConnector>
                <causalConnector id="onEndStopStart">

                    <simpleCondition role="onEnd"/>
                    <compoundAction operator="seq">
                        <simpleAction max="unbounded" role="stop"
qualifier="seq"/>
                        <simpleAction max="unbounded" role="start"
qualifier="seq"/>
                    </compoundAction>
                </causalConnector>
```

```
            </connectorBase>
        </head>
        <body>
            <port component="video" id="entryVideo"/>
            <port component="info" id="entry"/>
            <media descriptor="backgroundDesc" id="background"
src="resources/images/background.jpg"/>
            <media descriptor="videoDesc" id="video"
src="resources/video/video.mp4">
                <property name="bounds"/>
                <area begin="360s" id="startMenu"/>
            </media>
            <link xconnector="onEndStopStart">
                <bind role="onEnd" component="video"/>
                <bind role="stop" component="video"/>
                <bind role="start" component="video"/>
            </link>
            <media descriptor="infoDescriptor" id="info"
src="resources/images/info.png"/>
            <media id="settings" type="application/x-ginga-settings">
                <property name="stage" value="intro"/>
                <property name="interactivity" value="OFF"/>
                <property name="service.currentKeyMaster" value="1"/>
            </media>
            <switch id="interactivity">
                <bindRule constituent="startInteractivity"
rule="interactivityRule"/>
            <switch id="startInteractivity">
                <bindRule constituent="introContext" rule="introRule"/>
```

```
                        <bindRule constituent="menuContext" rule="menurule"/>
                        <context id="introContext" refer="introNCL#body"/>
                        <context id="menuContext">
                                <port id="p0" component="application"/>
                                <port id="p1" component="menu"/>
                                <port id="p2" component="despliegue"/>
                                <media id="application" src="main.lua"
type="application/x-ginga-NCLua">
                                        <property name="buttonPressed"/>
                                        <property name="menuChange"/>
                                        <property name="despliegueChange"/>
                                </media>
                                <link xconnector="onEndAttributionStopSetStart">
                                        <bind component="application"
interface="menuChange" role="onEndAttribution"/>
                                        <bind component="menu" role="stop"/>
                                        <bind component="application"
interface="menuChange" role="getValue"/>
                                        <linkParam name="var" value="$getValue"/>
                                        <bind component="menu" interface="src"
role="set"/>
                                        <bind component="menu" role="start"/>
                                </link>
                                <link xconnector="onEndAttributionStopSetStart">
                                        <bind component="application"
interface="delpliegueChange" role="onEndAttribution"/>
                                        <bind component="despliegue" role="stop"/>
                                        <bind component="application"
interface="despliegueChange" role="getValue"/>
                                        <linkParam name="var" value="$getValue"/>
```

```
                                        <bind component="despliegue" interface="src"
role="set"/>
                                        <bind component="despliegue" role="start"/>
                                </link>
                                <link xconnector="onKeySelectionSet">
                                        <bind component="application"
role="onSelection">
                                                <bindParam name="keyCode"
value="CURSOR_RIGHT"/>
                                </link>
                                <link xconnector="onKeySelectionSet">
                                        <bind component="application"
role="onSelection">
                                                <bindParam name="keyCode"
value="CURSOR_LEFT"/>
                                        </bind>
                                        <bind component="application"
interface="buttonPressed" role="set">
                                                <bindParam name="varSet"
value="CURSOR_LEFT"/>
                                        </bind>
                                </link>
                                <link xconnector="onKeySelectionSet">
                                        <bind component="application"
role="onSelection">
                                                <bindParam name="keyCode"
                                <link xconnector="onKeySelectionSet">
                                        <bind component="application"
role="onSelection">
                                                <bindParam name="keyCode"
value="CURSOR_DOWN"/>
```

```
                                        </bind>
                                        <bind component="application"
interface="buttonPressed" role="set">
                                                <bindParam name="varSet"
value="CURSOR_DOWN"/>
                                        </bind>
                                </link>
                        </context>
                </switch>
                </switch>
                <link xconnector="onBeginStopSetStart">
                        <bind component="video" interface="startMenu" role="onBegin"/>
                        <bind component="interactivity" role="stop"/>
                        <bind component="settings" interface="stage" role="set">
                                <bindParam name="var" value="menu"/>
                        </bind>
                        <bind component="interactivity" role="start"/>
                </link>
                <link xconnector="onKeySelectionStopSetStart">
                        <bind component="background" role="onSelection">
                                <bindParam name="keyCode" value="ENTER"/>
                        </bind>
                        <bind component="video" interface="bounds" role="set">
                                <bindParam name="var" value="0%,0%,100%,100%"/>
                        </bind>
                        <bind component="settings" interface="interactivity" role="set">
                                <bindParam name="var" value="OFF"/>
                        </bind>
```

```
                <bind component="info" role="start"/>
            </link>
            <link xconnector="onKeySelectionStopSetStart">
                <bind component="info" role="onSelection">
                    <bindParam name="keyCode" value="ENTER"/>
                </bind>
                <bind component="info" role="stop"/>
                <bind component="video" interface="bounds" role="set">
                    <bindParam name="var" value="42%, 3%, 55%, 55%"/>
                </bind>
                <bind component="settings" interface="interactivity" role="set">
                    <bindParam name="var" value="ON"/>
                </bind>
                <bind component="background" role="start"/>
                <bind component="interactivity" role="start"/>
            </link>
        </body>
</ncl>
```

Anexo 3 main.lua

```
local menuIndex = 0

local despliegueIndex = 0

function handler(evt)

	if evt.type == 'attribution' and  evt.name == 'buttonPressed' then

		if evt.value == "UP" and despliegueIndex ~= 0 then

			despliegueIndex = (despliegueIndex - 1) % 3

			dispatchEvent("despliegueChange",(despliegueIndex+1)..".png")

		end

		if evt.value == "DOWN" and despliegueIndex ~= 2 then

			despliegueIndex = (despliegueIndex + 1) % 3

			dispatchEvent("despliegueChange",(despliegueIndex+1)..".png")

		end

		if evt.value == "LEFT" and menuIndex ~= 0 then

			menuIndex = (menuIndex - 1) % 4

			dispatchEvent("menuChange",(menuIndex+1)..".png")

		end

		if evt.value == "RIGHT" and menuIndex ~= 3 then

			menuIndex = (menuIndex + 1) % 4

			dispatchEvent("menuChange",(menuIndex+1)..".png")

		end

	end

end

function attributionEventDispatch(action, property, value)

	return   event.post('out',{class = "ncl", type = "attribution", name = property,
action=action, value=value})

end
```

```
function dispatchEvent(property, value)
        attributionEventDispatch('start', property, value)
        attributionEventDispatch('stop',  property, value)
end
event.register(handler)
```

CAPITULO IV

Anexo I Clase TCP

```
---Librería para conexión TCP
--Fuente: <a href="http://www.telemidia.puc-
rio.br/~francisco/nclua/index.html">Tutorial de NCLua</a>
--Declaración de módulos, funciones globales, y constantes

local _G, coroutine, event, assert, pairs, type, print
    = _G, coroutine, event, assert, pairs, type, print
local s_sub = string.sub
module 'tcp'

–-uso de conexiones múltiples y manejo de las mismas

local CONNECTIONS = {}
local current = function ()
    return assert(CONNECTIONS[assert(coroutine.running())])
end
local resume = function (co, ...)
    assert(coroutine.status(co) == 'suspended')
    assert(coroutine.resume(co, ...))
    if coroutine.status(co) == 'dead' then
       CONNECTIONS[co] = nil
    end
end

function handler (evt)
    if evt.class ~= 'tcp' then return end
    if evt.type == 'connect' then
        for co, t in pairs(CONNECTIONS) do
            if (t.waiting == 'connect') and
               (t.host == evt.host) and (t.port == evt.port)
then
                t.connection = evt.connection
                t.waiting = nil
                resume(co)
                break
            end
        end
        return
    end

    if evt.type == 'disconnect' then
        for co, t in pairs(CONNECTIONS) do
            if t.waiting and
               (t.connection == evt.connection) then
                t.waiting = nil
                resume(co, nil, 'disconnected')
            end
        end
        return
    end

     if evt.type == 'data' then
```

```
        for co, t in pairs(CONNECTIONS) do
            if (t.waiting == 'data') and
            (t.connection == evt.connection) then
                resume(co, evt.value)
            end
        end
        return
    end
end
event.register(handler)

function execute (f, ...)
    resume(coroutine.create(f), ...)
    print("finalizado tcp.execute")
end

function connect (host, port)
    local t = {
        host    = host,
        port    = port,
        waiting = 'connect'
    }
    CONNECTIONS[coroutine.running()] = t

    event.post {
        class = 'tcp',
        type  = 'connect',
        host  = host,
        port  = port,
    }

    return coroutine.yield()
end

function disconnect ()
    local t = current()
    event.post {
        class      = 'tcp',
        type       = 'disconnect',
        connection = assert(t.connection),
    }
end

function send (value)
    local t = current()
    event.post {
        class      = 'tcp',
        type       = 'data',
        connection = assert(t.connection),
        value      = value,
    }
end


function receive (pattern)
    pattern = pattern or '' -- TODO: '*l'/number
```

```
    local t = current()
    t.waiting = 'data'
    t.pattern = pattern

    if s_sub(pattern, 1, 2) ~= '*a' then
        return coroutine.yield()
    end

    local all = ''
    while true do
        local ret = coroutine.yield()
        if ret then
            all = all .. ret
        else
            return all
        end
    end
end
```