Sistema de Monitorización de Calidad de Televisión Digital Descentralizado

César Schroeder, Federico Severo, Santiago Umpierrez

Mayo, 2015

Proyecto de grado dirigido por Profesor Eduardo Grampín y Profesor Javier Baliosian

Montevideo, Uruguay

Índice

Índice de Figuras

Índice de Archivos

Glosario

AAC Advanced Audio Coding es un formato de audio basado en un algoritmo de compresión con pérdida, y así obtener el mayor grado de compresión posible, resultando en un archivo de salida que suena lo más parecido posible al original [77]. 21

ANATEL Agencia Nacional de Telecomunicaciones. 15, 20

Apagón Analógico Cese de las emisiones analógicas de los operadores de televisión. Si bien se tenía el año 2012 como fecha límite de implementación para todos los países, en algunos aún no se ha llevado a cabo [2]. 15

ARIB Association of Radio industries and Business, es una organización de estandarización de Japón. Está a cargo de la promoción del uso eficiente del espectro de frecuencias [79] . 15

BML Broadcast Markup Language, o BML, es un estándar basado en XML desarrollado por JARIB (Japan's Association of Radio Industries and Businesses) como una especificación de broadcasting de datos para la televisión digital [72]. 20

BTS Broadcast Transport Stream, resultado del múltiplex de todos los programas emitidos por un radiodifusor. 24–27, 39, 42, 45, 49, 50, 53, 56, 58, 60, 61

códec Códec es la abreviatura de codificador-decodificador. Es el software o hardware o una combinación de ambos, que es capaz de transformar un archivo con un flujo de datos (stream) o una señal [87] plural. 31

definición de audio No existe una definición estándar de definición de audio sino que es un término asociado al marketing. Sin embargo, es usualmente utilizado para describir la frecuencia del audio. Por ejemplo, audio de alta definición se corresponde con el audio que tenga una frecuencia mayor que la percibida por el oído humano (20kHz) [68] [69]. 15

definición de video La definición de un video es una característica de un video que es definida por tres aspectos claves: La resolución de la imagen, generalmente identificada la componente vertical; el método de escaneo (progresivo o entrelazado); y el número de cuadros por segundo. Por ejemplo, una posible definición sería 720p60 que indica una resolución de 1280×720 con un método de escaneo progresivo a 60 cuadros por segundo [65] [66]. 15

DVB-T Digital Video Broadcasting - Terrestrial es el estándar para la transmisión de televisión digital terrestre creado por la organización europea Digital Video Broadcasting. 30

EPG Electronic Program Guide contiene la información referente a la programación planificada. 22

ES Elementary Stream, usualmente es audio o video codificado y solo está compuesto de un tipo de información. 25, 27, 39, 51, 52, 58, 61

GINGA Middleware utilizado para servicios de televisión digital terrestre del sistema japonÃ©s-brasilero ISDB-Tb. 27

H.262 Es un estándar de compresión de video desarrollado y mantenido conjuntamente por Video Coding Experts Group (VCEG) y Moving Picture Experts Group (MPEG). Es el utilizado por ISDB-T. 21

H.264 Método y formato para la compresión de video utilizado en el proceso de conversión de video digital, a un formato de menor tamaño para ser almacenado o transmitido. Es el utilizado por ISDB-Tb (Ver proceso de Codificación 2.2.3.1) . 21

HD Del inglés High Definition (Alta Definición) . 22

ISDB-T Integrated Services Digital Broadcasting - Terrestrial (ISDB-T) es el más avanzado estándar internacional en difusión de Televisión Digital Terrestre (TDT) desarrollado por Japón 2.2.1. 15, 20, 21

ISDB-Tb ISDB-Tb es la modificación del estándar ISDB-T japonés que propuso la ANATEL de Brasil 2.2.2. 15, 27, 30

log Un log es un registro de eventos durante un rango de tiempo en particular. Es similar a una bitácora [?] plural. 41, 51, 52, 59

middleware Capa de software que se asiste a aplicaciones en la interacción con otras aplicaciones, paquetes de programas, redes, y/o sistemas operativos. En el caso de la televisión digital, un middleware consiste en intérpretes de lenguajes y librerías de funciones que permiten la ejecución de aplicaciones en el receptor [76]. 20, 27

MPEG Moving Picture Experts Group, es la designación para un grupo de estándares de codificación de audio y video. 25, 34

MPEG-2 Es la designación para un grupo de estándares de codificación de audio y video acordado por MPEG (grupo de expertos en imágenes en movimiento), y publicados como estándar ISO 13818. MPEG-2 es por lo general usado para codificar audio y video para señales de transmisión, que incluyen Televisión digital terrestre, por satélite o cable [78]. 20

MPEG-4 Es un método para la compresión digital de audio y video. Fue introducido a finales de 1998 y designado como un estándar para un grupo de formatos de codificación de audio, video y las tecnologías relacionadas acordadas por la ISO |IEC Moving Picture Experts Group (MPEG) (ISO |IEC JTC1 |SC29/WG11), formalmente estándar ISO |IEC 14496 - Codificación de objetos audiovisuales [70]. 20, 21

One-Seg One-Seg es el nombre de un servicio de broadcasting que apunta a receptores con pantallas de tamaño reducido como teléfonos celulares [75]. 22

outlier En estadística, un outlier es una observación que es numéricamente distante del resto de los datos. Los valores estadísticos obtenidos a partir de datos que incluyen outliers pueden resultar engañosos. 39

PID Packet Identifier, campo de 13 bits que identifica a qué flujo elemental pertenece un determinado paquete TS. 24–27, 39, 47, 48, 55, 60

PSI Program Specific Information, es información que se agrega en al Transport Stream para indicar qué servicios y programas lo componen. 24, 25, 27

SAD Sum of Absolutes Differences. Es un indicador de la cantidad de movimiento de un video. 41, 42, 49, 81

SD Del inglés Standard Definition (Definición estándar) . 22

TDT Televisión Digital Terrestre. 20

TS Transport Stream, es un formato contenedor para la transmisión y almacenado de contenidos multimedia. Esta compuesto de paquetes TS que encapsulan Elementary Streams. 25–27, 34, 39, 60

Capítulo 1

Introducción

1.1 Motivación

La TV Digital, además de permitir mayor definición de video y definición de audio que la televisión analógica, también puede transmitir varias señales en un mismo canal. También permite a los televidentes interactuar con la pantalla utilizando el control remoto, sea para consultas de grillas de programación de los canales o ejecución de aplicaciones que utilicen Internet como vía de retorno de datos hacia el emisor. [3]

En Uruguay se está empezando generalizar la televisión digital y, como el número de países que decidieron ejecutar el denominado Apagón Analógico ha crecido en los últimos años, empieza a ser necesario ampliar los campos de investigación sobre la televisión digital.

Si bien la televisión digital asegura poder transmitir señales audiovisuales con mayor definición que la señal de televisión analógica, no hay garantías sobre la calidad percibida por los televidentes respecto al audio y video emitidos por las empresas televisoras. Por esta razón, resulta necesario ejercer algún control sobre estas emisiones. Para poder realizarlo, sería idóneo contar con un sistema automatizado que monitorice la calidad de las emisiones de cada proveedor.

1.2 Problema

En el marco de la implementación de la Televisión Digital en Uruguay, se planteó la problemática de desarrollar un sistema que permita determinar y registrar la calidad de servicio de emisiones de televisión digital. En Uruguay se optó por la norma ISDB-Tb, basada en la norma ISDB-T de origen japonés desarrollada por el consorcio ARIB (*Association of Radio industries and Business*) y modificada por la ANATEL (Agencia Nacional de Telecomunicaciones) de Brasil; esta modificación es la que se eligió finalmente para su implementación en Uruguay.

La televisión digital, a diferencia de la analógica, posibilita la compresión de los contenidos originales de audio y video, permitiendo así aumentar el número de canales emitidos y haciendo posible la emisión de contenidos en alta definición.

Las emisiones televisivas se transmiten, en general, a través de radiodifusión (en inglés *broadcasting*). La radiodifusión, es el proceso por el cual estas emisiones son recibidas por el público general a través de la utilización ondas electromagnéticas como medio de transporte.

En el proceso de trasmisión de contenidos digitales se debe comprimir y empaquetar los originales antes de ser emitidos. Tanto en la etapa de compresión como durante la trasmisión se producen degradaciones sobre los mismos que luego podrían ser percibidas por un televidente. La degradación producida durante la etapa de compresión es producto de los algoritmos utilizados y la degradación en la etapa de transmisión se debe a: atenuación de la señal de radio, interferencias del medio, ecos, etc.

Estudios recientes se han centrado en poder medir la calidad de las transmisiones televisivas y, de forma más general, la calidad de video tal como es percibida por el espectador. La calidad de video es un término a veces confuso, ya que depende de muchos factores incluyendo de quién la evalúe, es decir, es un término subjetivo. De esta manera, se han creado modelos que permiten simular de forma aproximada la opinión de las personas que ven un video. Estos modelos están en constante evolución y calibración debido a que dependen de factores subjetivos y a que, como es un tema reciente de investigación, surgen nuevos indicadores a tener en cuenta. En general, los modelos dependen de diferentes indicadores de calidad percibida y estos se pueden definir como información extraída de los audios y videos que puede ser relevante a la hora de obtener una evaluación de calidad. Este proceso de extracción de información suele poder ser automatizable ya que usualmente consiste en analizar el audio o video a través de un cierto algoritmo.

Es importante aclarar que el resultado de dicha extracción de información podría depender de cada muestra. Como se mencionó anteriormente, en distintas ubicaciones geográficas se podrían producir degradaciones durante la transmisión del contenido y de esta manera tener distintas estimaciones de calidad para distintas ubicaciones geográficas.

Conociendo la existencia de estas formas de estimación de la calidad de las emisiones, las empresas televisoras podrían estar interesadas en tomar medidas frente a una estimación de calidad de resultado malo o bajo en una cierta ubicación geográfica. Ejemplos de medidas a tomar podrían ser: verificar que realmente la calidad sea mala o baja, aumentar la potencia de señal, o enviar soporte técnico a la zona. De esta manera podría ser interesante también para las empresas poder tener un sistema que les provea dicha estimación.

1.3 Objetivos del proyecto

1.3.1 Objetivo general

Debido a lo mencionado anteriormente, y a que actualmente no se conoce en nuestro país un sistema que provea dicha información a las empresas televisoras ni a

un ente regulador, en el presente proyecto analizaremos aspectos relativos a este problema y desarrollaremos un prototipo capaz de dar soporte a la obtención de indicadores útiles para la estimación de la calidad de las emisiones de los servicios de televisión digital. Nos centraremos en la infraestructura de software que permitirá ejecutar algoritmos de obtención de indicadores útiles para distintos modelos de calidad. Una vez recolectados los indicadores, se podría ejecutar un modelo de calidad que los utilice.

Como prueba de concepto de este proyecto, se utilizará un modelo particular para probar dicha infraestructura y así ver con más precisión la utilidad del sistema. Es decir se configurará el sistema para obtener los indicadores requeridos por dicho modelo y efectivamente realizar el cómputo de calidad.

1.3.2 Objetivos específicos

1.3.2.1 Objetivos primarios

Se plantea el objetivo de diseñar y construir un sistema que permita dar soporte a la obtención de indicadores (o parámetros de modelos) sobre un conjunto de canales de televisión en una serie de ubicaciones geográficas.

Es importante también que el sistema tenga la capacidad de almacenar un histórico de los indicadores y de las estimaciones de calidad, y en el caso de detectar alguna anomalía, contar con una evidencia de la muestra en la que fue detectada.

Debido a que los modelos de calidad se encuentran en constante evolución, la infraestructura deberá permitir configurar los indicadores que se obtienen proveyéndole a la misma de los algoritmos necesarios para obtenerlos.

1.3.2.2 Objetivos secundarios

Además del cómputo de modelos de calidad sobre los programas emitidos, el sistema deberá también admitir la medición de otros parámetros independientes de un programa específico, como lo es la potencia de señal. También deberá ser posible la incorporación de medidas arbitrarias provistas por otros sensores, como por ejemplo factores climáticos.

Actualmente el Uruguay se encuentra en un proceso en donde los canales de televisión se encuentran haciendo pruebas y variando sus emisiones, y se prevé en breve que nuevos canales comiencen también a emitir. Por esta razón, el sistema deberá admitir también la configuración de canales y emisiones a monitorizar.

1.4 Estructura del documento

En el capítulo 2 se introducen conceptos previos requeridos para el entendimiento del problema y la solución. Luego, en el capítulo 3, se presenta un estado del arte en el que se comentan soluciones existentes al problema y se identifica el aporte. Los capítulos que le siguen plantean la solución, la arquitectura, el diseño, el hardware y software utilizado para las pruebas, las decisiones del equipamiento

y los resultados de las pruebas realizadas. Finalmente, analizando el trabajo realizado, se presentan las conclusiones y el trabajo a futuro.

Capítulo 2

Fundamentos

Este capítulo define e introduce algunos conceptos necesarios para la comprensión de la problemática y la solución planteada. Inicia con la definición de Televisión digital, identificando la norma con la que se va a trabajar, algunas características de la misma y precisando en la metodología de compresión, envío y recepción de la transmisión televisiva.

2.1 Televisión Digital

La televisión digital (DTV en inglés) es un conjunto de tecnologías de transmisión y recepción de contenido audiovisual a través de señales digitales. A diferencia de la televisión tradicional (analógica), esta codifica sus señales de forma binaria, permitiendo un mejor aprovechamiento del ancho de banda mediante el aplicado de compresión de video[1].

Por otro lado, la señal de televisión digital se transmite en un ancho de banda de 6MHz a una tasa de 19.6 Mbps, en la que es posible enviar más programas y contenido por el mismo canal. De esta manera es posible tener varias emisiones en un mismo canal utilizando formatos diferentes. La capacidad de transmisión presenta así una forma de uso y distribución flexible, en la que las televisoras pueden escoger qué tipo de señal incluir en el canal.

En la siguiente figura se muestran distintas formas de utilización del canal.

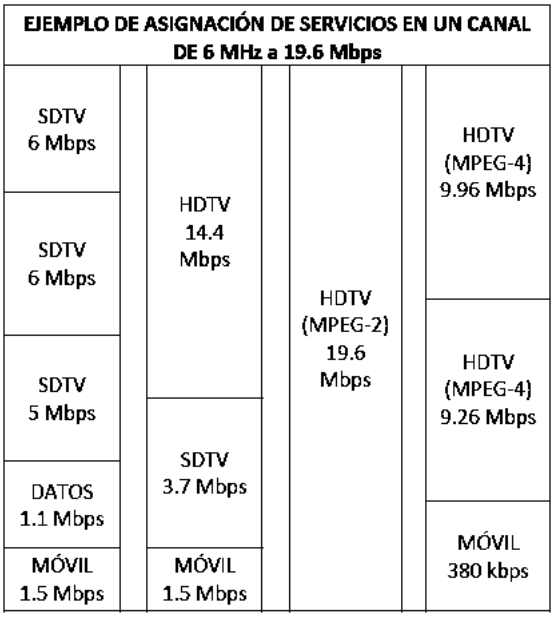

Figura 2.1: Utilización de Canal [21].

2.2 ISDB-T / ISDB-Tb

2.2.1 ISDB-T

Integrated Services Digital Broadcasting - Terrestrial (ISDB-T) es el más avanzado estándar internacional en difusión de Televisión Digital Terrestre (TDT) desarrollado por Japón. Fue introducido en dicho país en Diciembre del 2003 y el número de países que lo adoptan ha ido aumentando gradualmente con el reconocimiento de sus ventajas tecnológicas.

2.2.2 ISDB-Tb

Como se mencionó anteriormente, ISDB-Tb es la modificación del estándar ISDB-T japonés que propuso la ANATEL de Brasil. Esta nueva norma resultante exige la utilización de codificación de video en MPEG-4 (sustituyendo MPEG-2) y GINGA[10] como middleware (sustituyendo BML) para aplicaciones de televisión digital interactiva.

2.2.3 Características de ISDB-T - ISDB-Tb

ISDB-T presenta muchas ventajas técnicas que se basan principalmente en su estructura la cual está conformada por subsistemas entre los cuales se resaltan: el de transmisión, codificación, multiplexación, recepción, interactividad, guía de operación y seguridad. En lo concerniente a este proyecto, los sistemas de interés son el de codificación, el de multiplexación y el de recepción.

En la siguiente imagen se puede apreciar la comunicación de los subsistemas mencionados en la norma brasilera.

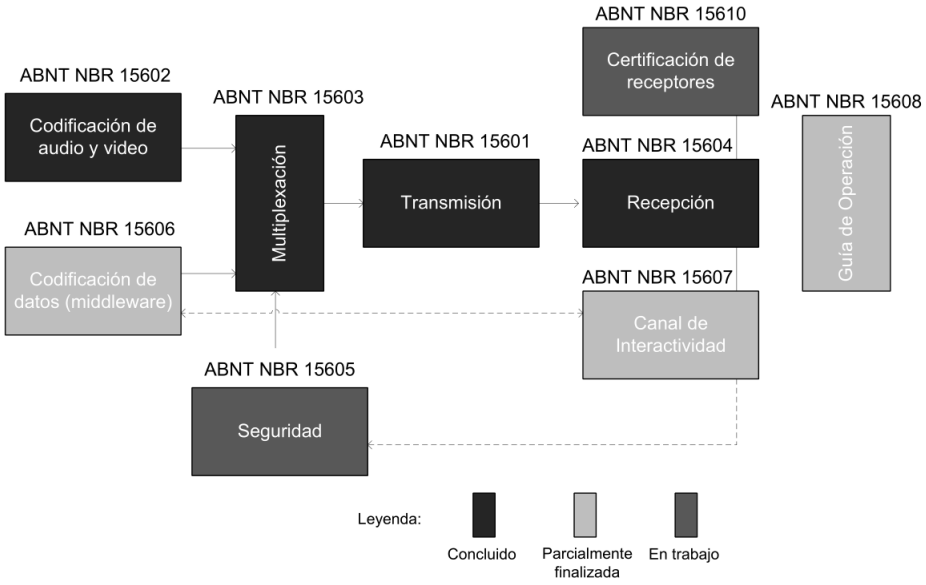

Figura 2.2: Subsistemas ISDB-Tb [21].

2.2.3.1 Codificación

El subsistema de codificación tiene como objetivo mejorar la tasa de transmisión de la señal de audio y video. Implementa la codificación H.264 que mejora la interpretación de los movimientos de la imagen, con respecto H.262 utilizado por ISDB-T y se apoya en la codificación MPEG-4 AAC para minimizar la degradación y el ruido.

H.264
Es un método y formato para la compresión de video utilizado en el proceso de conversión de video digital, a un formato de menor tamaño para ser almacenado o transmitido. La compresión (o codificación) de video es una tecnología esencial para aplicaciones como televisión digital, video DVD, *mobile* TV, videoconferencias y *streaming* de video en Internet. La estandarización de compresión de video lograría la interoperabilidad de los productos de diferentes proveedores como codificadores, decodificadores y medios de almacenamiento. Un codificador convierte video en un formato comprimido y un decodificador convierte video comprimido de vuelta al formato original (antes de la codificación) o al formato necesario para ser reproducido en el dispositivo [71].

Típicamente se utiliza para compresión con perdida en el sentido matemático estricto, a pesar de que a veces la pérdida es imperceptible. Aun así, es posible crear

codificaciones sin pérdida [22]. H.264 provee una sintaxis claramente definida para la representación de video comprimido y la información relacionada.

En el nivel más alto, una secuencia H.264 consiste en una serie de paquetes o "Network Adaptation Layer Units" (NAL Units o NALUs). Pueden incluir conjuntos de parámetros conteniendo parámetros claves que son utilizados por el decodificador para decodificar correctamente los datos de video y los "slices", que son *frames* codificados o parte de ellos. En el siguiente nivel, un *slice* representa un *frame* de video o parte de él, y consiste en un número de macrobloques codificados, cada uno conteniendo datos comprimidos correspondientes a un bloque de 16×16 píxeles. En el nivel más bajo, un macrobloque contiene información que describe el método particular utilizado para la compresión del mismo, información de predicción como los vectores de movimiento codificados entre otros [23].

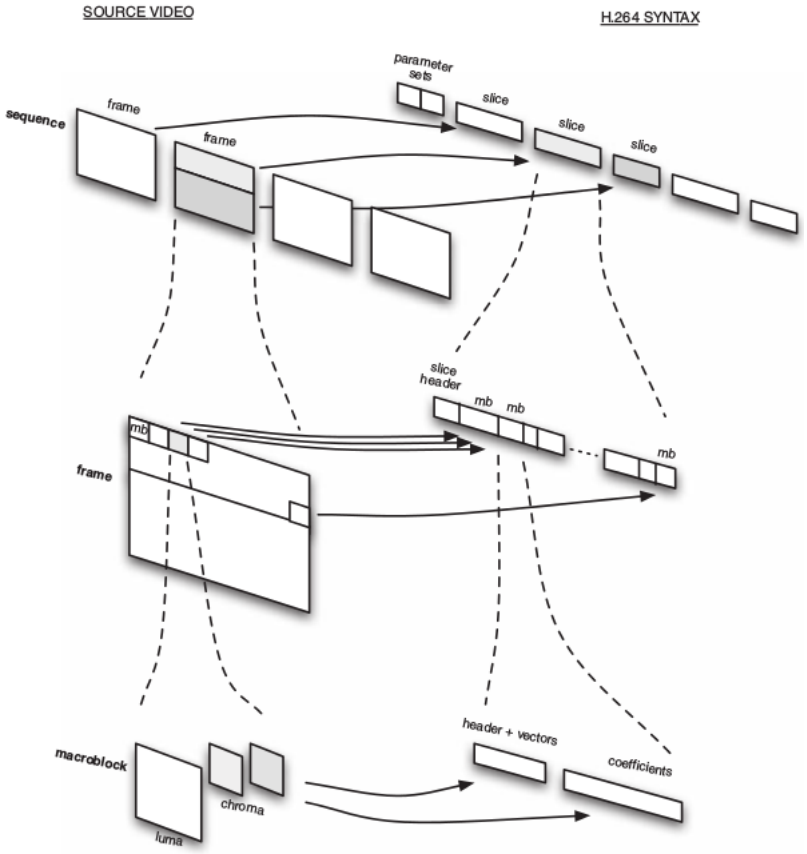

Figura 2.3: H.264 syntax: overview [23].

2.2.3.2 Multiplexación

Este subsistema recibe las señales codificadas de audio y video (HD - *High Definition*, SD - *Standard Definition*, One-Seg), datos (EPG , interactividad) y ac-

tualización de los receptores vía aire para encapsularlas y enviarlas en una trama denominada **BTS** (*Broadcast Transport Stream*) la cual posee una tasa fija de 32.507936 Mbps.

Durante esta etapa, cada uno de los elementos mencionados en la etapa anterior (denominados **ES** - *Elementary Stream*), es empaquetado en paquetes **TS** (*Transport Stream*) de 204 bytes que se distribuyen de la siguiente manera:

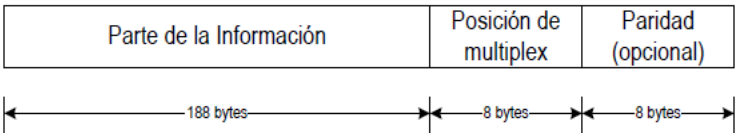

Figura 2.4: Distribución Transport Stream [21].

Paquete TS
Un paquete TS de 188 bytes está compuesto por un *header* (cabezal) de longitud fija de 4 *bytes* y un *payload* (carga) de 188 bytes.

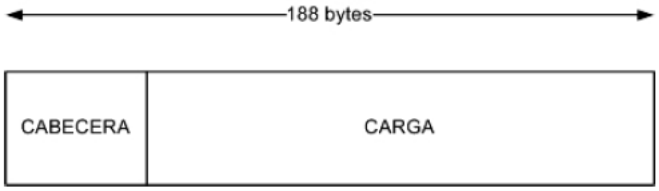

Figura 2.5: Contenido de un paquete TS.

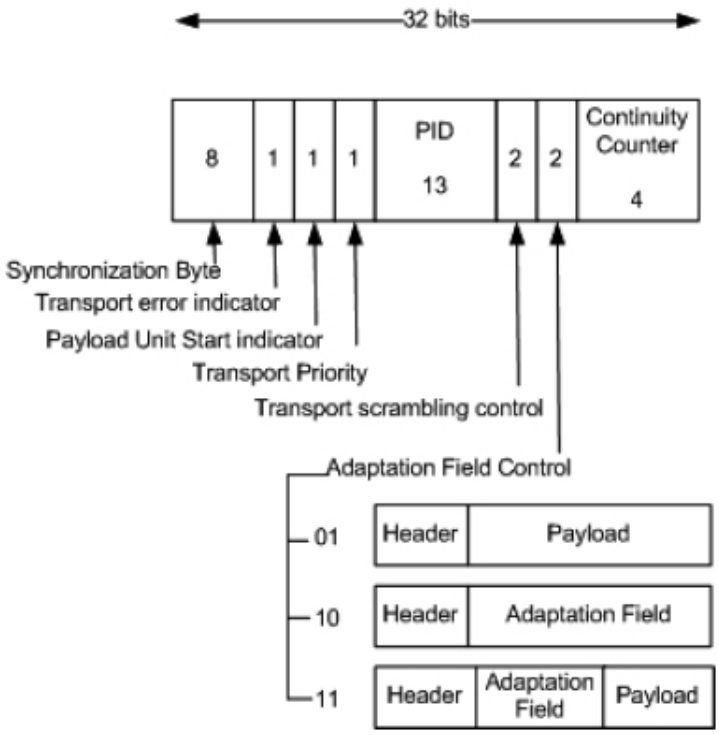

Figura 2.6: Cabecera de paquete TS.

El campo *Synchronization Byte* siempre contiene el valor 0x47, aunque este valor también puede aparecer en otros campos, es utilizado para identificar el comienzo de cada paquete TS. El hecho de que aparezca cada 188 *bytes* simplifica la tarea del decodificador para identificar el comienzo de estos paquetes de transporte.

Debido a que un BTS puede contener múltiples programas diferentes, y a su vez cada programa contiene múltiples flujos elementales, el campo *Packet Identifier* (PID) indica a que flujo elemental pertenece el paquete TS. El campo *Continuity Counter* es un contador que incrementa en 1 entre sucesivos paquetes TS correspondientes al mismo flujo elemental.

Identificación de Paquetes TS
Para que el decodificador pueda reconstruir completamente un programa a partir de los paquetes TS que componen el BTS, es necesario incluir información adicional que relacione los números de PID entre sí, esto es imprescindible para identificar cuáles son los flujos elementales que componen un servicio determinado.

Para esto se cuenta con Información de Servicio (SI) e Información Específica de Programa (PSI), esta información se incluye en paquetes TS adicionales agregados al BTS y enviados periódicamente.

2.2.3.3 Service Information (SI)

Está compuesta por las siguientes 4 tablas.

Network Information Table (NIT): Posee número de PID reservado 0x0010 y contiene información definida por el radiodifusor, como frecuencias del canal, detalles de modulación, etc.

Service Description Table (SDT): Posee número de PID reservado 0x011 y contiene nombres de los servicios emitidos, nombre del proveedor, y otros parámetros asociados a cada servicio dentro del múliplex.

Event Information Table (EIT): Posee número de PID reservado 0x012 y contiene información sobrenombres de programas y tiempos de comienzo.

Time and Date Table (TDT): Posee número de PID 0x014 y contiene información sobre fecha y hora actual para configurar la hora interna del receptor.

Una vez formados los paquetes TS se multiplexan, como se mencionó anteriormente, formando el BTS. Dentro de la trama BTS, el multiplexor combina diversos contenidos de entrada y los señaliza de forma tal que el receptor pueda decodificar los flujos de audio, video y datos. Para poder realizar esta tarea se utilizan las tablas PSI (*Program Specific Information*) y SI (*Service Information*).

La siguiente figura presenta un ejemplo de una trama BTS con sus respectivas tablas PSI:

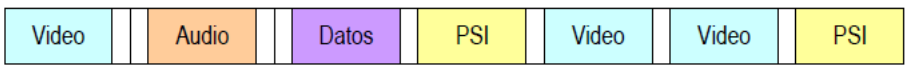

Figura 2.7: Ejemplo de trama BTS [21].

Dentro de las tablas que integran el conjunto de tablas PSI encontramos la tabla PAT (*Program Asociated Table*). En dicha tabla se le asigna un PID (*Packet Identifier*) a los paquetes TS para cada servicio en el multiplexor. La funcionalidad de dicha tabla radica en la reconstrucción de los distintos ES a través de los TS que se obtienen al demultiplexar los BTS con el PID de la tabla.

2.2.3.4 Program Specific Information (PSI)

Definido por MPEG, está compuesta por las siguientes tablas.

Program Association Table (PAT): Posee número de PID 0x0000 y contiene una lista con todos los servicios disponibles en el BTS. Cada servicio aparece junto con el número de PID de los paquetes TS que contienen la contienen las tablas PMT, estas tablas indican los programas que conforman el servicio.

Program Map Table (PMT): Posee número de PID entre 0x0002 y 0x001F y se dispone de una tabla PMT por cada servicio presente en el BTS. Cada PMT especifica los números de PID de los programas (flujos elementales) que componen el servicio.

La siguiente figura esquematiza el funcionamiento de las tablas PAT y PMT.

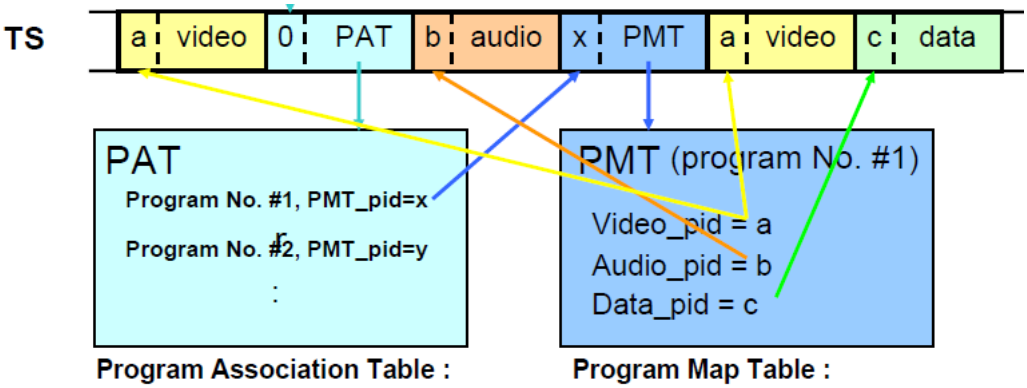

Figura 2.8: Tablas PAT y PMT.

Conditional Access Table (CAT): Esta tabla debe estar presente si al menos uno de los programas del múltiplex es de acceso condicional (está cifrado). Se transporta esta tabla en paquetes TS con número de PID 0x0001.

La siguiente figura resume el proceso de multiplexación:

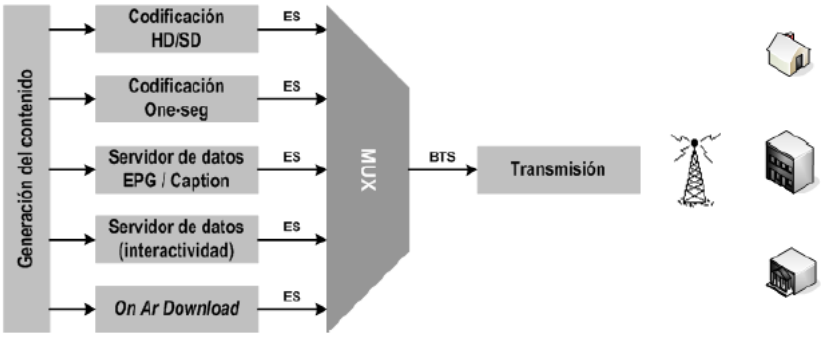

Figura 2.9: Proceso de multiplexación [21].

Luego de multiplexados los TS, se modulan para ser transmitidos.

2.2.3.5 Recepción

Este subsistema se encarga de la demodulación, decodificación y demultiplexación de la señal recibida. Existen diversos tipos de dispositivos que realizan este pro-

ceso, entre los que podemos destacar los televisores con receptor integrado, los Set-Top-Box y los Dongles USB.

2.2.3.6 Estructura de los datos

Es importante tener presente la estructura en la que se transmiten y reciben los datos, ya que es con la que se trabajará durante el proyecto. Resumiendo los datos obtenidos en la sección anterior, un receptor recibe las tramas BTS, las cuales a través de las tablas PSI y los PID demultiplexa los TS y los une para reconstruir los ES. Los ES son los elementos que analizaremos para obtener parámetros de calidad y los BTS son los que se analizarán para obtener datos referentes a la pérdida de paquetes.

2.2.3.7 Interactividad en ISDB-T

El estándar ISDB-Tb provee mecanismos para agregar dentro de la señal de un proveedor de servicios de televisión aplicaciones interactivas que permiten al usuario recibir más datos sobre la programación del proveedor o acceder a servicios y juegos.

Hay dos tipos de aplicaciones, interactivas locales o interactivas completas. Las locales usan sólo una vía de comunicación que es del proveedor al receptor. Su utilidad es principalmente la de brindar información extra sobre la programación al televidente.

Además, hay otro tipo de aplicaciones interactivas que hacen uso de un canal de retorno del televidente hacia el emisor por medio de Internet. Para que esto funcione, es necesario tener el receptor de TV Digital conectado a Internet. Esta comunicación de dos vías permite aplicaciones verdaderamente interactivas y el acceso a servicios más complejos mediante el uso del control remoto como pueden ser pagos o votaciones en encuestas.

Para que las aplicaciones puedan ejecutarse en el receptor, es necesario un middleware que traduzca la información recibida a través de la señal de TV Digital en datos y ejecutables. En el estándar ISDB-Tb, el middleware utilizado es GINGA desarrollado en TeleMídia Lab - PUC-Rio[74].

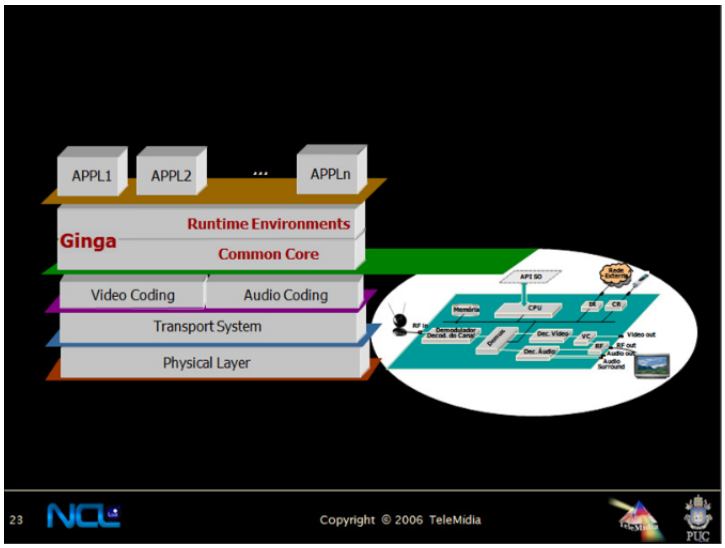

Figura 2.10: GINGA.

Ginga[10] está basado en NCL (Nested Context Language) , un lenguaje declarativo que permite crear aplicaciones multimedia. Utiliza LUA[64] como lenguaje de *scripting* pero además permite la ejecución de código Java.

Las aplicaciones NCL se dividen en etapas o escenas anidadas, cada una de ellas cuenta con diferentes objetos, que pueden ser imágenes, videos, sonidos, documentos HTML (Hyper-Text Markup Language), *scripts* en LUA[64] o código Java. NCL es el guion que relaciona a todos estos objetos en el espacio y tiempo.

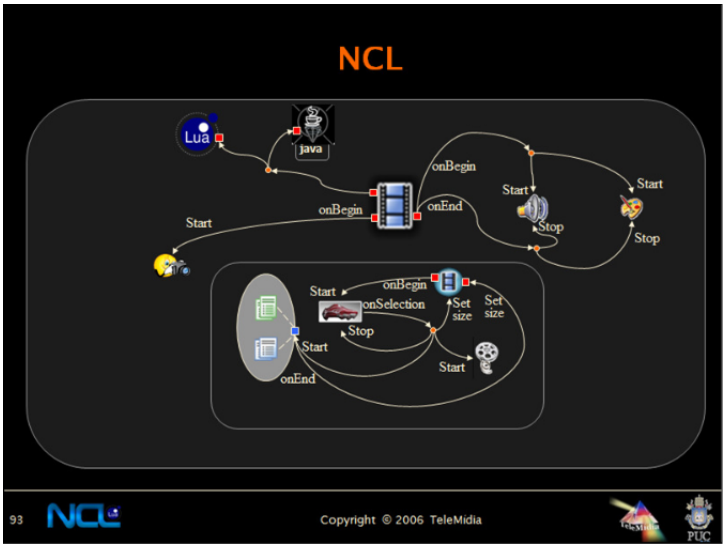

Figura 2.11: NCL.

2.3 Redes de Sensores

Como los indicadores de calidad de video pueden variar según la ubicación geográfica debido a las pérdidas de propagación o degradación de la señal, es importante que haya varios nodos distribuidos en distintas zonas. Estos nodos se encargarían de recolectar indicadores de calidad de video de las emisiones que allí reciban. Cada nodo podría medir todos o un conjunto de los parámetros necesarios por los modelos, donde estos últimos podrían trabajar de forma conjunta para recolectar los parámetros necesarios para computar un cierto modelo de calidad. Esta idea se asemeja en gran medida a lo que se conoce como una "Red de Sensores".

Una red de sensores, es una red en la que cada nodo consta de un dispositivo con sensores y un sistema de transmisión/recepción de datos. Cada nodo es capaz de procesar una cantidad limitada de datos, pero coordinando la información obtenida por un número importante de los mismos, se logra medir un medio con gran detalle.

Se puede describir a una red de sensores como un grupo de nodos que colaboran en una tarea común.[4] Dependiendo de la densidad del despliegue y la coordinación entre los nodos, este tipo de redes llevan sus tareas con más o menos precisión.

Actualmente las redes de sensores son distribuidas e inalámbricas, permitiendo una mayor cantidad de nodos logrando la localización de un fenómeno con mayor facilidad.[5]

2.4 Sistemas de Monitorización

Cada nodo de una red de sensores obtiene un conjunto de indicadores de calidad de video, pero ellos mismos por sí solos pueden no tener suficiente significado por lo que resulta importante a veces verlos como conjunto o incluso en su evolución en el tiempo. Esta información podría ser centralizada y visualizada en un sistema de información que provea de una interfaz gráfica. En este aspecto, podría referirse a este sistema como un "Sistema de Monitorización".

Los sistemas de monitorización son sistemas orientados a la obtención de información del entorno donde se despliegan con el objetivo de analizarla posteriormente. En función de este análisis se tomarán las medidas pertinentes. La información del entorno es obtenida a través redes de sensores.

Este tipo de sistema permite hacer un seguimiento de los valores recopilados por todos los sensores que luego, a través de una interfaz gráfica, es posible visualizarlos. En general constan también de gráficas y estadísticas que facilitan el análisis. De esta manera es deseable que un sistema de monitorización esté funcionando de forma ininterrumpida [6]. Es importante destacar que son sistemas pasivos, ya que solo recolectan los datos y los muestran, y el análisis y control es realizado por personal competente [7].

2.5 Sistemas embebidos

Cada nodo sería un sistema que, a través de infraestructura de hardware y software, toma medidas sobre el medio y ejecuta algoritmos para la obtención de los indicadores. Estos nodos podrían bien ser computadoras o simplemente dispositivos orientados únicamente a la función de obtener dichos indicadores. Visto de esta manera, cada nodo podría ser diseñado especialmente para la función descrita prescindiendo de hardware y software que no sería utilizado. Cada nodo en sí, podría constituir un sistema embebido diseñado especialmente para el propósito de obtener indicadores de calidad de video.

Un sistema embebido es un sistema de computación diseñado para realizar una o algunas pocas funciones predefinidas, usualmente con requerimientos muy específicos. Debido a que están orientados a tareas específicas, los ingenieros pueden optimizarlos y/o reducir su costo. La mayoría de sus componentes se encuentran incluidos en la placa base y en general no tienen el aspecto generalmente asociado a una computadora. Algunos ejemplos de sistemas embebidos pueden ser taxímetros, cajeros automáticos, impresoras, calculadoras, entre otros.

Este campo de investigación tiene un gran potencial debido a que combina dos factores. En primer lugar que el diseñador del sistema tiene control sobre ambos, el diseño del hardware y el del software, a diferencia de las computadoras de propósito general. En segundo lugar, construir un sistema embebido involucra múltiples disciplinas, como arquitectura de computadoras (arquitectura y microarquitectura de procesadores, diseño de sistema de memoria), compiladores, planificación y sistema operativo o sistemas de tiempo real. Combinar estos dos factores nos permite romper las barreras entre los campos mencionados, permitiendo así una optimización más grande que al ser consideradas de forma independiente [8].

En general son programados en lenguaje *assembler* o también C o C++; y en algunos casos, cuando el tiempo de respuesta no es un factor crítico pueden utilizarse lenguajes interpretados como JAVA [9].

El propósito de los sistemas embebidos es lograr un buen rendimiento. Algunos de ellos poseen requerimientos de tiempo real y, si las tareas no se realizan en el tiempo estipulado, el sistema podría fallar y perjudicar al usuario.

2.6 Receptores de TV digital

Los nodos entonces necesitan una forma de obtener la muestra a procesar para obtener los indicadores. Para dicho propósito es necesario contar con una manera de grabar las emisiones de video en formato ISDB-Tb.

Investigando receptores, percibimos que existe mayor oferta comercial de dispositivos compatibles con otros estándares de televisión, como DVB-T y luego de una observación más detenida se puede ver que muchas de las distintas marcas son en realidad el mismo producto reetiquetado. Además la gran mayoría de estos dispositivos proveen compatibilidad solamente bajo sistemas operativos Windows.

Luego de una búsqueda exhaustiva encontramos en plaza el sintonizador ISDB-Tb USB MyGica S870 Full-Segment[51] compatible con Windows y Linux.

2.7 Modelos de calidad

Recientemente se ha intentado obtener modelos objetivos y algoritmos para predecir la calidad de video percibida en diferentes escenarios. Se han propuesto numerosos modelos de estimación para predecir el nivel de calidad que percibiría un observador promedio, en base a parámetros extraídos de muestras de video y audio.

Estos modelos se clasifican en tres grupos, los modelos Full-Reference son aquellos modelos en donde para "medir" la calidad se comparan parámetros del contenido original con los obtenidos en el contenido comprimido y degradado por la trasmisión. Por otro lado tenemos los modelos Reduced-Reference que incluyen un conjunto reducido de datos sobre la fuente original y finalmente están los modelos No-Reference, que solamente se basan en el análisis del contenido potencialmente degradado, es decir lo que efectivamente recibe un televidente en su hogar.

Los modelos paramétricos, predicen la calidad de video percibida basándose en un número reducido de parámetros, que tienen una relación con el proceso de codificación, el contenido del video e información de la red. Estos modelos típicamente se basan en una fórmula matemática que representa la estimación de la calidad percibida de video en función de los distintos parámetros. Son fáciles de implementar debido a que no necesitan la muestra de video original. La estimación de la calidad puede ser obtenida muy rápidamente ya que es el resultado de la fórmula matemática [17]. Notar que la carga estaría en los procesos de obtención de parámetros.

Es decir, los modelos de calidad pueden depender de distintos parámetros así como de requerir el video original.

2.7.1 MOS - Mean Opinion Score

Es un test frecuentemente utilizado para obtener el punto de vista de los usuarios humanos respecto a la calidad de un servicio. Es frecuentemente utilizado para estimar la calidad de redes de telefonía o *Voice over IP* (VoIP).

En multimedia, especialmente cuando los códecs son utilizados para comprimir los datos de manera de cumplir con los requisitos de ancho de banda de la red, el MOS provee una estimación numérica de la calidad percibida por los usuarios, expresada como un número entre 1 y 5, donde 1 es la menor calidad y 5 la mayor.

La siguiente tabla expresa los posibles valores de calidad [16]:

5	Excelente
4	Buena
3	Aceptable
2	Pobre
1	Mala

Capítulo 3

Estado del arte

En el siguiente capítulo se analizarán otros trabajos vinculados a la temática.

3.1 Proyecto VQI

En el Proyecto VQI[19], ademas de un modelo de calidad No-Reference, se propone la arquitectura de un sistema capaz de computar su modelo de calidad a muestras de televisión de 10 segundos tomadas una vez por minuto. El sistema propuesto se descompone en módulos que cumplen las tareas de tomar muestras, computar el modelo sobre la muestra, enviar los datos de calidad recolectados a un servidor central que posee una base de datos y una aplicación web en donde consultar los datos obtenidos.

Para validar el modelo propuesto se creo una base de datos de videos de referencia a los que se les ha introducido degradación, alterando valores de los parámetros definidos como indicadores de calidad. Luego se sometió estos videos a evaluación subjetiva de calidad por parte de un conjunto de evaluadores humanos y se compararon los resultados de calidad medidos por el modelo y los puntajes otorgados por los evaluadores.

En etapas posteriores del proyecto se han realizado pruebas sobre muestras de televisión digital, obteniendo resultados satisfactorios.

En la sección Modelo de Calidad Candidato se describe en mayor detalle el modelo propuesto.

3.2 Hardware dedicado

Varios fabricantes, entre ellos Rohde&Schwarz[11], han desarrollado equipos capaces de realizar mediciones en tiempo real sobre aspectos técnicos de emisiones de televisión digital.

Por ejemplo, el R&S ETL[12] nos permite medir MER(*Modulation Error Ratio*), BER (*Bit Error Rate*), PER (*Packet Error Ratio*) y *bitrate*, entre otros.

3.3 Proyecto analizador de TS

En la Escuela Politécnica del Ejército de Ecuador se ha llevado a cabo un proyecto en donde se diseña una aplicación capaz de generar reportes a partir de caraterísticas extraidas de una muestra de TS con codificación en MPEG [86].

Este reporte incluye la detección de programas y servicios y análisis de la cabecera de los paquetes TS.

3.4 Otras utilidades disponibles

El paquete TStools, cuenta con un conjunto amplio de utilidades para el procesamiento de muestras de TS. Cuenta con aplicaciones capaces de extraer datos sobre cantidad de paquetes, datos sobre composición de programas y flujos, y extracción de programas específicos.

El paquete LinuxTV dvb-apps[43] contiene aplicaciones para la interacción con sintonizadores digitales. En particular cuenta con la aplicación tzap, utilizada para la sintonización de frecuencias de televisión digital.

DVBSnoop[42] es un analizador de flujos MPEG, funciona de forma similar a un *sniffer* y es utilizado para la visualización o depuración de información sobre flujos MPEG.

FFMpeg[20] es un conjunto de aplicaciones de software libre capaz de decodificar, transcodificar y aplicar filtros a contenidos audiovisuales.

3.5 Redes de sensores

Las redes de sensores han demostrado ser aplicables para tomar medidas de medios de distinta índole. En nuestro país existen ejemplos y pruebas piloto aplicados a actividades agrícolas ([80], [81]). También existen proyectos orientados a otros tipos de aplicaciones como la estructural [82].

3.6 Conclusiones

La abstracción propuesta en este proyecto permite la independencia y el cómputo simultáneo de múltiples modelos de calidad aplicados a los distintos programas emitidos por los radiodifusores.

El objetivo de proveer una plataforma para la obtención de parámetros y cómputo de modelos, permite la independencia del hardware dedicado existente, dándole la posibilidad a un desarrollador interesado de definir nuevos indicadores y la forma de calcularlos.

Debido a la naturaleza de la problemática planteada, es necesario analizar paquetes TS y realizar procesamiento sobre la señal recibida en los distintos puntos

geográficos. En este aspecto, las utilidades mencionadas proveen una ayuda sustancial. Además dicha naturaleza, propone a este proyecto una nueva aplicación del concepto de redes de sensores extendiendo así su alcance.

Capítulo 4

Requisitos de la solución

En este capítulo se comentarán los requisitos impuestos sobre la solución que se abarcarán. Debido a la naturaleza del problema, y a lo mencionado sobre sistemas de sensores y de monitorización, el mismo se puede resolver en dos sistemas. Un sistema que cumplirá el rol de sensor y otro que se corresponderá con el sistema de monitorización que recibirá y desplegará la información.

4.1 Sistema sensor (Nodo Remoto)

4.1.1 Requisitos Funcionales

Se describirán las funcionalidades que el sistema deberá proveer.

1. El sistema deberá poder configurar canales.
2. El sistema deberá poder grabar la emisión de Televisión Digital del canal que se le indique.
3. El sistema deberá poder configurar la duración de la muestra a grabar.
4. El sistema deberá poder computar cualquier cantidad de algoritmos de complejidad diversa sobre una muestra grabada.
5. El sistema deberá poder extraer los resultados de los algoritmos.
6. El sistema deberá poder enviar los resultados de los algoritmos al sistema de monitorización.
7. Una muestra sobre la que se ejecutan algoritmos de procesamiento, deberá ser almacenada en disco y permanecer accesible al sistema de monitorización por el mayor tiempo posible.

4.1.2 Requisitos no Funcionales

Se describirán los requisitos no funcionales sobre el sistema.

1. El sistema deberá poder ejecutar los algoritmos definidos sobre una muestra en un lapso de tiempo menor a la duración de la misma.

4.2 Sistema de monitorización

4.2.1 Requisitos Funcionales

Se describirán las funcionalidades que el sistema deberá proveer.

1. El sistema deberá poder reconocer cada nodo por separado.
2. El sistema deberá poder almacenar un histórico de los valores de los indicadores de calidad para cada Nodo Remoto que le envía información.
3. El sistema deberá poder admitir la configuración y ejecución de un algoritmo sobre los valores de los indicadores que podría generar un nuevo indicador.
4. El sistema deberá poder exhibir en pantalla los valores de cada nodo de cada indicador de calidad.
5. El sistema deberá poder solicitar una muestra que esté disponible en un Nodo Remoto a demanda.

4.2.2 Requisitos no Funcionales

Se describirán los requisitos no funcionales sobre el sistema.

1. La interfaz de usuario deberá ser orientada a los datos.

Capítulo 5

Solución propuesta

En la siguiente sección se describirán los puntos abarcados por la solución con una breve explicación de cómo se resuelven. Además se presentará el modelo candidato que fue utilizado de ejemplo para la configuración y para las pruebas.

5.1 Obtención de muestras

Previo a la medición de parámetros, necesitamos obtener muestras de los programas emitidos por los canales.

A través de Receptores USB es posible capturar los BTS, concatenarlos y guardarlos en un archivo. Como se mencionó anteriormente, los BTS tienen TS de todos los ES que se transmiten y por tanto, a través del demultiplexado y concatenación de los TS identificados por el número de PID, se pueden obtener los distintos fragmentos de ES de los cuales se obtienen los parámetros.

Debido a que existen modelos de calidad que contemplan la calidad afectada por propagación, es necesario tomar muestras en distintos puntos geográficos. Para esto se contará con un conjunto de estaciones distribuidas para la obtención de muestras.

De esta forma una muestra quedará identificada por el canal al que pertenece, por la ubicación y por el tiempo de inicio de la grabación.

5.2 Modelo de calidad candidato

En [17] se plantean una serie de modelos paramétricos para la estimación de calidad de video percibida y se realiza una comparación en base a parámetros utilizados, y precisión en términos de RMSE (*Root Mean Square Error*) y porcentaje de *outliers*.

Decidimos experimentar con el modelo de calidad de video propuesto por el Proyecto VQI [18] principalmente por dos razones: por un lado, en el estudio comparativo citado obtenía muy buenos resultado, y por otro lado, al ser un proyecto que fue realizado en la misma institución, facilitaba la comunicación

frente a dudas conceptuales o de implementación del mismo.

Este modelo consiste en un modelo de calidad de video No-Reference que se descompone en calidad de video en origen y calidad de video afectada por propagación.

La calidad de video en origen (Ic) está determinada por la siguiente expresión:

$$I_c = 4\left(1 - \frac{1}{1 + \left(\frac{\left(\frac{hl}{720 \times 586}\right)^{-a_1} b}{c_1 s + c_3}\right)^{c_4 s + c_3}}\right) \tag{5.1}$$

En donde los parámetros recibidos son los siguientes:
h : Alto de la pantalla en píxeles
l : Ancho de la pantalla en píxeles
b : Bitrate
s : SAD medio por píxel
$a_1, c_1, c_3, c_4 \text{y} c_6 \text{son constantes}$

La calidad afectada por propagación está determinada por la siguiente expresión:

$$I_p = \frac{1}{1 + kp_w} \tag{5.2}$$

$$p_w = \frac{r_a}{r} \tag{5.3}$$

En donde los parámetros recibidos son los siguientes:
r : Cantidad de slices presentes en el video
r_a : Cantidad de slices afectados

Teniendo las expresiones anteriores, el MOS se obtiene utilizando la siguiente expresión:

$$V_q = 1 + I_c I_p \tag{5.4}$$

El resultado de la expresión anterior representa el puntaje entre 0 y 5 que un observador promedio le otorgaría al clip de video en cuestión. Se experimentará con este modelo en particular, pero el sistema a construir admitirá también otros modelos de calidad de video, así como modelos de calidad de audio e interactividad.

5.2.1 Obtención de parámetros

Los parámetros requeridos por el modelo anterior se pueden obtener de la siguiente manera:

5.2.1.1 Slices totales y afectados

El grupo de trabajo del Proyecto VQI modificó el código del FFMpeg [20], de modo de agregar en su salida información adicional sobre *slices*.

La obtención de la proporción de los *slices* afectados se obtiene computando un algoritmo sobre dicha salida.

Si invocamos esta aplicación con el comando

```
./ffmpeg_g -debug pict -i [Video] -vcodec rawvideo -f null /dev/null
```

Donde `[Video]` es una muestra de video a analizar, obtenemos en pantalla un log que contiene la información necesaria para el calculo de estos parámetros.

El algoritmo consiste en recrear todos los cuadros de video en base a los *slices* y propagando los errores detectados en función del tipo de *slice* (I, P o B). Se implementó este algoritmo en lenguaje C.

5.2.1.2 Resolución y bitrate

La resolución de pantalla y el *bitrate* se puede obtener utilizando la aplicación mediainfo[24] con el comando:

```
mediainfo [Video]
```

El comando anterior, junto a otros datos, retorna en pantalla los valores deseados.

5.2.1.3 SAD medio por pixel

Para la obtención del SAD promedio por píxel nos basamos en el trabajo del Proyecto VQI. A partir de una muestra de video descomprimida (en raw), se ejecuta un script AviSynth [26] mediante la aplicación VirtualDub [25]. En el script Avisynth se indica que el proceso exhiba información sobre movimiento en el video, retornando un archivo que deberá ser parseado y procesado para obtener el valor buscado.

Este grupo de trabajo reimplementó en lenguaje C, el algoritmo de *parsing* (análisis de entrada) y procesado para incluir soporte a *multithreading* (multi-hilos) y además incrementar la velocidad de cálculo, reduciendo este a una fracción del tiempo insumido por el original.

Esta modificación permitió que todo el procesamiento requerido para obtener el parámetro SAD, pueda ser realizado en un tiempo inferior a la duración de una muestra.

5.2.2 Observaciones

Al medir los parámetros relacionados a la calidad de video en origen, se obtendrán los mismos valores para cada ubicación en donde no se produzcan perdidas de paquetes. Por esta razón se propone medir estos parámetros en una sola ubicación (en donde se pueda asumir que la perdida es despreciable) y utilizar estos mismos valores para todas las ubicaciones que se deseen monitorizar.

De esta forma se podría monitorizar la calidad afectada por propagación en forma distribuida y la calidad de video en origen en forma centralizada.

Esto reduciría significativamente los costos de hardware para las estaciones de monitorización debido a que la obtención del parámetro SAD implica un procesamiento considerablemente mayor al necesario para la obtención de la proporción de *slices* afectados por propagación.

5.3 Medición de parámetros

Para la medición de un parámetro de un programa emitido por un canal, se debe en primer lugar extraer dicho programa del archivo de BTSs y aplicar sobre este un conjunto de pasos hasta obtener el valor deseado.

Este proceso puede ser, en algunos casos, demasiado pesado como para realizarse centralizadamente en un único nodo. Esto se debe a que el sistema deberá soportar la medición de parámetros de varios canales, en varias ubicaciones y para cada canal se extraerán parámetros para un conjunto de programas.

Si elegimos realizar esta tarea en un único nodo, será necesario también transportar las muestras obtenidas en cada estación hacia este nodo, lo que implica también una mayor carga sobre la red subyacente.

Consideramos como mejor alternativa, realizar el cómputo de parámetros en las propias estaciones distribuidas. Es decir, cada estación extraerá los parámetros de los programas de cada una de sus muestras.

Bajo esta estrategia, será necesario transportar únicamente los valores de los parámetros medidos y una muestra de grabación solamente en caso que amerite retornar una evidencia.

Se identifica también un conjunto de parámetros independientes de programas, como potencia de señal, BER, factores climáticos y parámetros obtenidos a partir del archivo de BTSs sin la necesidad del demultiplexado de un programa específico. Estos parámetros no están contemplados por los actuales modelos de calidad, pero el sistema a construir podrá obtenerlos dejando la puerta abierta a trabajos a futuro que contemplen este tipo de mediciones en la evaluación de la calidad de servicio.

5.4 Cómputo de Modelos

Una vez obtenidos los parámetros medidos sobre un programa, se debe computar las funciones de calidad. Este paso determina si se está bajo una condición de servicio anormal que amerite la obtención de evidencias.

Si este cómputo se realiza en las estaciones distribuidas, cada una deberá ser capaz de obtener todos los parámetros necesarios. Como se menciona anteriormente, esta estrategia no siempre resulta viable en términos de la potencia del hardware requerido por las estaciones.

Consideramos como mejor alternativa, realizar el cómputo de los modelos de calidad centralizadamente. Es decir que tendremos un nodo para el cómputo de las funciones de calidad a partir de los parámetros individuales medidos por las estaciones.

Esto nos permite paralelizar en distintas estaciones la obtención de parámetros, incluso para un mismo modelo. Además permite la abstracción de modelos en las estaciones, es decir, las mismas trabajarán a nivel de parámetro y no de modelo.

Esta decisión permite también, a la hora de determinar la pertinencia de obtención de evidencia, tomar decisiones en base a la realidad de más de una estación teniendo un punto de vista más amplio.

5.5 Flexibilidad de configuración

Como se menciona anteriormente, deberá ser posible para cada canal y programa deseado, la configuración de qué parámetros se medirán y qué modelos de calidad se computarán.

Con esto, la configuración de parámetros, canales y programas será utilizada por las estaciones distribuidas y la configuración de modelos será utilizada por el nodo que computará los modelos de calidad.

Capítulo 6

Arquitectura propuesta

En este capítulo se describirá la arquitectura de la solución, comprendiendo los módulos de la misma, una descripción del alcance en base a estos, y la distribución *geográfica* de los módulos.

6.1 Módulo Generador

Este módulo se encargará de recibir y almacenar los archivos de BTSs para un conjunto de canales configurados, para esto este módulo interactuará con sintonizadores digitales.

Sintonizará un conjunto de canales y grabará continuamente partiendo las muestras a intervalos fijos. Una vez obtenida una nueva muestra, se notificará al módulo Procesador indicando a que canal pertenece la muestra y el tiempo de inicio de grabación.

6.2 Módulo Procesador

Se encargará de la extracción de parámetros a partir de los archivos de BTSs grabadas por el Módulo Generador.

Para esto, contará para cada programa y canal deseado, una secuencia de pasos a seguir para obtener cada uno de los parámetros deseados.

Una vez obtenido un parámetro, se notificará al Módulo Recolector indicando a que tiempo, programa y canal pertenece el parámetro.

6.3 Módulo Recolector

El Módulo Recolector, se encargará de tomar las mediciones realizadas por el Módulo Procesador y transportarlas al Sistema Centralizado.

6.4 Sistema Centralizado

A partir de valores de parámetros, los almacenará, computará las funciones de calidad de los modelos y poseerá una interfaz para exhibir los datos obtenidos.

También deberá poseer un mecanismo que permita detectar situaciones anormales y en caso de que amerite, obtener las muestras de grabación correspondientes.

6.5 Alcance del proyecto

En el presente trabajo, diseñaremos e implementaremos un prototipo de los módulos Generador, Procesador y Recolector. Adicionalmente, para la verificación de los otros módulos realizaremos un prototipo del Sistema Centralizado que almacenará y exhibirá los valores recibidos y calculados, se omitirán aspectos relativos a la detección y manejo de situaciones anormales.

6.6 Distribución

6.6.1 Estaciones remotas

Son un conjunto de equipos en donde ejecutarán los Módulos Generador, Procesador y Recolector. Cada módulo será un ejecutable independiente. Para la toma de muestras de canales deberán contar con sintonizadores digitales.

6.6.2 Nodo central de recolección de datos

Consiste de un nodo que ejecutará el Sistema Centralizado.

6.7 Diseño de la solución

6.7.1 Interfaces dentro de la estación

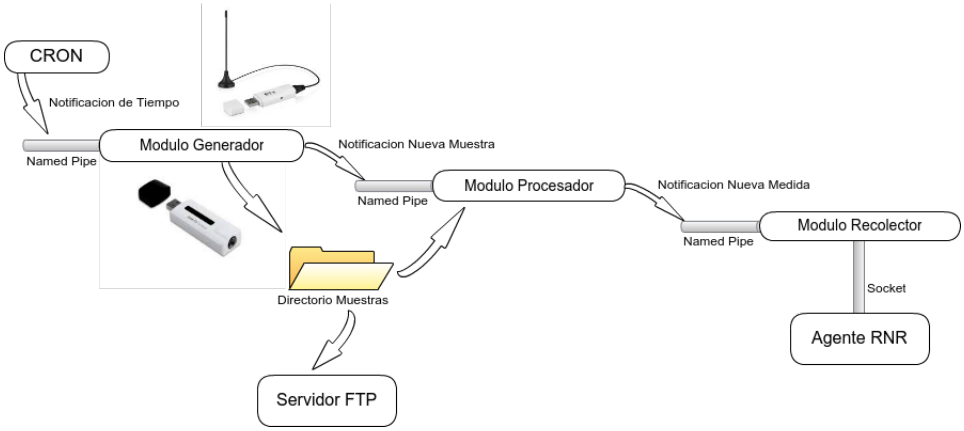

Figura 6.1: Interfaces de comunicación dentro de la estación.

Cada uno de los módulos dentro de la Estación, se comunican mediante *Named Pipes FIFO*.

Elegimos este mecanismo porque pueden ser abiertos por múltiples procesos, y al intercambiar datos a través de ellos el *Kernel* pasa la información de un proceso a otro, directamente sin escribir ni leer datos del *filesystem*. Esto evita la utilización del almacenamiento primario para el intercambio, agilizando significativamente el proceso.

Con el objetivo de que los tres módulos ejecuten independientemente, decidimos que los procesos que escriben datos a un pipe, lo hagan en modo lectura/escritura de modo de siempre lograr acceso (el modo escritura es bloqueante) y en caso de que no haya ningún proceso leyendo, la salida se descarte y el escritor no se bloquee ocupando espacio en memoria.

6.7.1.1 Generador-Procesador

Una vez obtenida una nueva muestra, el Módulo Generador escribirá en el pipe de notificaciones del Procesador, el nombre del archivo muestra grabado. Los nombres de los archivos de muestra siguen la siguiente estructura:

nombreCanal_aaaammddhhmmss.ts

En este nombre se incluye el nombre del canal al que pertenece la muestra y el tiempo de inicio de grabación, un ejemplo de archivo puede ser

Canal 10_20141208105820.ts

6.7.1.2 Procesador-Recolector

Al obtener el valor de un parámetro, para que sea enviado al Sistema Centralizado, se deberá escribir en el pipe donde lee el Módulo Recolector un string con el siguiente formato:

[aaaammddhhmmss|canal|programa|parámetro|valor]

aquí se indica la fecha y nombre de canal al que pertenece la muestra, junto con numero de PID del programa (si corresponde), nombre y el valor del parámetro medido. Un ejemplo de este string puede ser:

[20141121205230|TNU|501|bitratevideo|12005]

6.7.2 Interfaces con el Sistema Centralizado

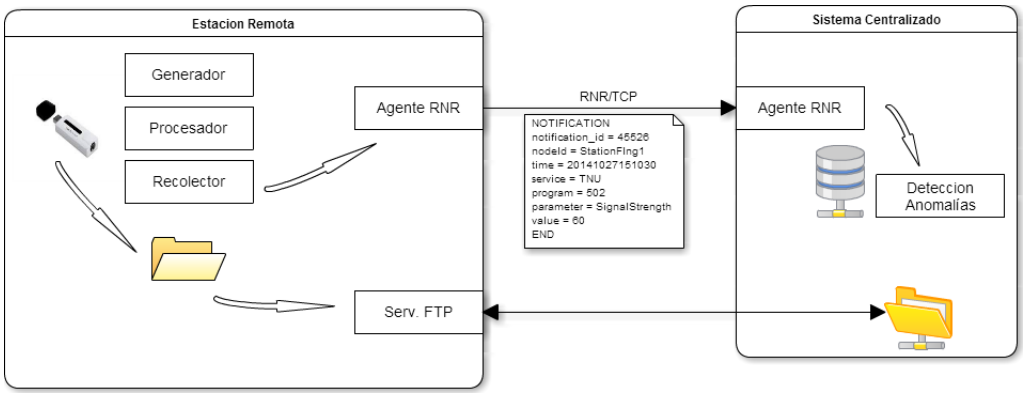

Figura 6.2: Interfaces de comunicación con el Sistema Centralizado.

Para el transporte de las mediciones de parámetros desde las Estaciones al Sistema Centralizado utilizamos el protocolo RNR [27].

Consiste en un bus de notificaciones guiado por publicaciones y suscripciones basadas en el contenido de los mensajes enviados sobre TCP. Cada nodo del sistema RNR ejecutará un agente y será este agente quien se encargue del transporte de los mensajes entre nodos según suscripciones.

Este protocolo fue sugerido por los tutores del proyecto y tras un análisis de factibilidad determinamos que era adecuado para la realidad del problema. Esta decisión se basó principalmente en la baja carga que implica tanto sobre la red, como de *overhead* de procesamiento en el envío y recepción de los mensajes, y por contar con una implementación accesible y pública del protocolo.

La estructura de las notificaciones elegida para el envío de parámetros es la siguiente:

Archivo 6.1: Notificación RNR Ejemplo

```
1 NOTIFICATION
2 Notification_id=9588
3 nodeId=Station1
4 time=20141117110630
5 channel=TNU
6 program=402
7 parameter=videoBitrate
8 value=9530
9 END
```

Aquí se indica la estación que genera la medida, el tiempo de inicio de grabación y canal al que pertenece la muestra, en caso corresponda se incluye el número de PID del programa y el valor del parámetro medido.

Para la obtención de evidencias desde las estaciones, el sistema central las solicitará a través de algún protocolo de transferencia de archivos. Para esto es necesario que en las estaciones ejecuten un servidor que implemente el protocolo y también es necesario definir una nomenclatura que nos permita identificar un archivo de muestra.

Para tomar una decisión sobre qué protocolo de transferencia de archivos utilizar, se investigaron los más comunes, HTTP [37] y FTP [36]. A través de pruebas basadas en la transferencia de archivos de diversos tamaños, y, considerando como factor decisivo el consumo de CPU, se determinó que, por una diferencia muy pequeña, FTP consume menos recursos en el aspecto mencionado (Ver A.1). Para las pruebas mencionadas, se utilizó el demonio FTPD [29] para FTP y el demonio Webfsd [28] para HTTP. Por otro lado FTP posee funcionalidades que agregan un mínimo de gestión a las estaciones, ya que permite borrar y agregar archivos a las estaciones de forma remota. De esta manera se decidió que FTP era la mejor opción en este caso.

6.7.3 Estimación de volumen de datos

Tanto las medidas de los parámetros por parte de las estaciones, como las evidencias a transportar hacia el Sistema Centralizado, implican tráfico de datos sobre la red.

En lo que respecta al transporte de evidencias, el volumen generado está vinculado a la cantidad de muestras solicitadas y al tamaño de cada muestra. Un archivo de muestra de 10 segundos conteniendo la grabación del archivo de BTSs de un canal ocupa entre 17 y 22 MB [1] que deberán ser enviados al Sistema Centralizado.

El volumen de datos sobre la red resultante del envío de parámetros, estará determinado por la cantidad de parámetros a medir en un intervalo de tiempo y el tamaño de cada mensaje. Si para el transporte de parámetros utilizamos notificaciones RNR siguiendo la estructura propuesta, cada mensaje ocupará aproximadamente 140 bytes sobre TCP [34]

Para calcular la cantidad de mensajes generados, supongamos que monitorizamos, en una única estación, solamente un programa de video de un canal de televisión. Supongamos también que las muestras se encuentran divididas en intervalos de 10 segundos y que medimos los parámetros necesarios para el modelo candidato incluyendo también la potencia de señal. Bajo las suposiciones anteriores se generarán cada 10 segundos, un mensaje por cada parámetro medido, es decir seis mensajes (*bitrate*, resolución H, resolución V, SAD, proporción *slices* afectados y potencia de señal). Debido a que los parámetros a medir, como los canales y emisiones a monitorizar es configurable y las estaciones podrán tener configuraciones distintas, se deberá aplicar un razonamiento análogo para determinar el volumen de datos sobre la red generados según la realizad configurada. Las estaciones deberán disponer de una ventana de tiempo para el almacenamiento de muestras, este tiempo está determinado por lo que tardaría el Sistema Central

[1] Estos valores fueron obtenidos de forma empírica a través del almacenamiento de emisiones de 10 segundos de los distintos canales de transmitidos en Uruguay. Esta diferencia posiblemente se deba al proceso de codificación que realiza cada uno de los emisores.

en recibir todos los parámetros requeridos por los modelos aplicados a emisiones de la estación, y el tiempo que tardaría en reconocer una situación anormal, que amerite la solicitud de muestras de evidencia.

Una muestra de 10 segundos, conteniendo el BTS completo (todos los programas) emitido por un canal de televisión, ocupa entre 17 y 22 MB dependiendo de los *bitrates* de los programas emitidos.

Entonces, por cada *Gigabyte* de almacenamiento destinado a muestras en las estaciones, se disponen siete minutos de grabación repartidos entre todos los canales monitorizados.

6.7.4 Sincronización entre estaciones

Al elegir una arquitectura distribuida, en donde se toman grabaciones de duración fija en distintas ubicaciones, se nos plantea el problema de la sincronización en el proceso de medición de calidad. Esto es necesario para que las estaciones evalúen la señal de una misma emisión y tener así la evaluación en las distintas zonas geográficas.

Ya que el proceso de obtención de parámetros comienza una vez obtenida una muestra, el problema se reduce a que los Módulos Generadores de todas las estaciones estén sincronizados.

Notar que el sincronismo es únicamente respecto a la grabación de muestras y no al procesamiento, ya que los nodos podrían recolectar parámetros diferentes. Esto se debe a que una vez sincronizados los grabadores, los nodos al comunicar los parámetros calculados indican en base a qué fragmento fue tomado y eso es lo que los relaciona a la hora de computar una función de calidad.

Para resolver este problema decidimos que cada ciclo de grabación del Módulo Generador sea iniciado por una notificación de tiempo, y que esta notificación esté sincronizada para todas las estaciones.

Dado que no existe conexión entre estaciones, pero deben estar sincronizados de modo de tomar las muestras al mismo tiempo, decidimos que cada estación utilice un servidor NTP [35] para que sus relojes de sistema mantengan la misma hora. La razón por la que se eligió este protocolo es por ser el más utilizado para este propósito.

Como NTP define una jerarquía de servidores, y éstos tiene un error (del orden de milisegundos) lineal en número de *hops* (saltos) de servidor, es importante destacar que solo se podrían sincronizar todas las estaciones si estas utilizan servidores que están en el mismo nivel de la jerarquía. Además, como las estaciones pueden ser definidas en una intranet, es deseable considerar ambos casos, que las terminales tengan o no Internet.

Si las estaciones no poseen conexión a Internet, se sincronizan con algún servidor local. Según [31] la carga de protocolo no es muy significativa y se puede afirmar que aún una computadora no muy potente puede soportar una cantidad

sustancial de clientes con una degradación mínima de los servicios de red. Esto nos permite concluir que un servidor NTP no redundaría en un costo elevado de hardware. De esta manera, contar con un conjunto de servidores NTP locales sería viable. Ahora bien, cuando todos los nodos tienen conexión a Internet, éstos pueden ser sincronizados contra servidores externos públicos, ya que éstos están preparados para soportar cargas muy grandes y permiten evitar los costos del servidor de NTP. Es interesante destacar que los retardos de la red podrían demorar el proceso de sincronización entre los terminales y el servidor. De este modo sería deseable además que los servidores NTP que se elijan, además de estar en el mismo nivel, estén próximos en *hops* de red. Una opción sería utilizar los servidores de NTP de Brasil publicados en [30] ya que serían cercanos y posiblemente estén sincronizados contra el mismo servidor o entre sí.

Una vez sincronizados los relojes de cada Estación, basta con configurar adecuadamente el Crontab [41] de modo que notifique al Módulo Generador en los intervalos deseados.

Es importante destacar que si bien la sincronización es vital para obtener las evaluaciones correctas, un error del orden de milisegundos (introducido por NTP) sería tolerable, debido a que, en muestras de duración de algunos segundos, no alteraría drásticamente los resultados.

6.7.5 Interpretación y plan de ejecución del Modelo Candidato

En la sección Modelo de calidad candidato se describen los pasos a seguir para la obtención de los parámetros requeridos por el modelo a partir de una muestra de video. En rasgos generales se observa que el proceso implica la ejecución de aplicaciones (mediante comandos de terminal) que consumen tanto el archivo original como otros archivos producidos durante el proceso. Un ejemplo de esto es el procesamiento del log producido por el FFMpeg para la obtención de la proporción de *slices* de video afectados.

Debido a que uno de los objetivos del proyecto es permitir que la obtención de parámetros sea configurable, en el presente trabajo se logró abstraer esta metodología mediante un lenguaje basado en tareas que pueden ser vinculadas entre sí. Una tarea se concibe como un ejecutable que consume y produce archivos y el lenguaje permite el encadenamiento de estas tareas mediante una nomenclatura en donde se describen entradas y salidas.

De esta forma logramos que sea posible crear estructuras arborescentes de tareas, en donde una vez ejecutado cada nodo, se dispondrán los archivos necesarios para la ejecución de sus nodos hijos. Esta estructura es expresada en un archivo de configuración que se describe con mayor detalle en la sección Módulo Procesador.

Para la obtención de parámetros de calidad a partir de una muestra de ES, se configura un árbol en donde su raíz consuma el ES en cuestión y sus hojas retornen el valor del parámetro de calidad deseado, luego el "camino a seguir" desde la raíz a cada una de las hojas quedará determinado por la cantidad de pasos

intermedios necesarios.

Con esto logramos mantener para cada ES obtenido (determinados por configuración) un árbol que ejecutará para obtener efectivamente todos los parámetros requeridos por todos los modelos de calidad a aplicar sobre el ES.

La imagen siguiente modela el proceso de obtención del parámetro necesario para el cómputo de la componente de calidad afectada por propagación.

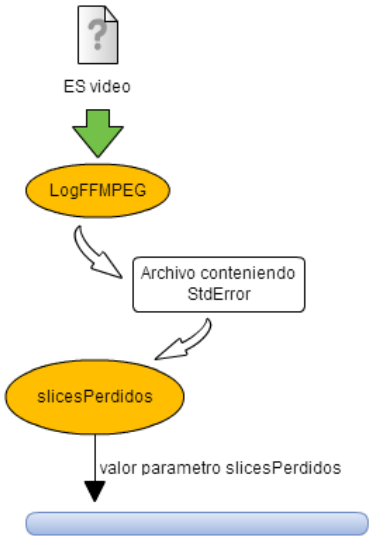

Figura 6.3: Modelado de calidad de video afectada por propagación.

Aquí se observan dos tareas en donde la primera (LogFFMPEG) consume el ES obtenido y produce un archivo de log, el cual es consumido por la segunda tarea (slicesPerdidos) para la obtención de la proporción de *slices* de video afectados.

La imagen siguiente modela el proceso de la obtención de los parámetros necesarios para el cómputo de la componente calidad de video en origen.

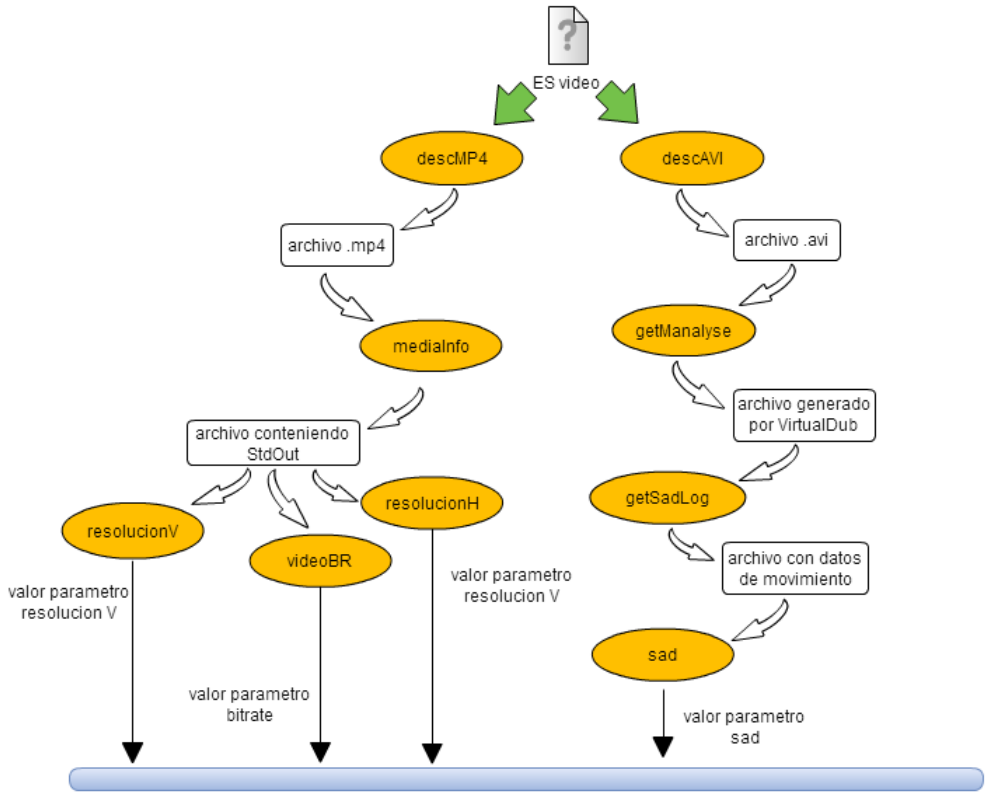

Figura 6.4: Modelado de calidad de video origen.

6.7.6 Módulo Generador

El rol principal de este módulo es la grabación continua de muestras de BTS de un conjunto de canales de televisión. En el proceso de la obtención de muestras se debe primero sintonizar cada canal, para esto se utiliza la aplicación Tzap [40]. Una vez sintonizado el canal deseado, se graba el BTS emitido con la aplicación DvbSnoop [42].

Para cada canal configurado, este módulo iniciará un hilo encargado del sintonizado, ejecutando la aplicación Tzap, que permanecerá activo durante toda la ejecución del módulo. Luego, en cada ciclo de grabación iniciará un hilo encargado de la grabación que ejecutará la aplicación DvbSnoop durante el tiempo de grabación configurado. De esta forma, por cada canal configurado se tendrán activos en todo momento dos hilos.

6.7.6.1 Configuraciones

A continuación se describirán las configuraciones que se realizarán sobre el Módulo Generador.

Configuración del módulo
Consiste en un archivo con configuraciones básicas para el módulo. Se indican aquí, la ubicación del archivo de canales, el directorio en donde se almacenarán temporalmente las muestras tomadas, la cantidad de espacio de almacenamiento destinado a muestras y las rutas de los pipes en donde recibirán las marcas de tiempo y donde se notificará al Módulo Procesador. En este archivo se incluirá también, el nombre de cada canal a muestrear junto con la ruta y numero de adaptador del Dongle a utilizar para cada canal.

Archivo 6.2: Archivo de Configuración del Generador

```
<?xml version="1.0" encoding="UTF-8"?>
<conf>
    <channelsFile>
        /home/user/SMTVD/Conf/canales.conf
    </channelsFile>
    <recordsPath>
        /home/user/SMTVD/Grabaciones/
    </recordsPath>
    <recordsQuota>100</recordsQuota>
    <sampleDuration>10</sampleDuration>
    <pipeOutNewRecord>
        /home/user/SMTVD/npNewRecord
    </pipeOutNewRecord>
    <pipeTimeFeed>
        /home/user/SMTVD/npTime
    </pipeTimeFeed>
    <Channels>
        <Channel name="TNU">
            <device value="/dev/dvb/adapter0"/>
            <adapterNum value="0"/>
        </Channel>
    </Channels>
</conf>
```

Donde:

channelsFile
Indica el la ubicación del archivo de canales.

recordsPath
Indica la ubicación en la que se guardarán las muestras.

recordsQuota
Indica el tamaño en megas destinado al almacenamiento de archivos.

sampleDuration
Indica la duración de la muestra a grabar.

pipeOutNewRecord
Indica el Named Pipe al que escribir el nombre de los archivos grabados (el que el procesador utiliza).

pipeTimeFeed
Indica el Named Pipe en el que se recibe la señal comenzar a grabar.

Channels
es donde se configuraran los canales a grabar a y los sintonizadores que se utilizaran para cada uno.

Channel
Indica un canal a grabar. El atributo nombre de este tag se deberá corresponder con uno de los nombres del archivo de canales que se describirá a continuación.

device
Indica el path de Linux del dispositivo que grabará el canal.

adapterNum
Indica el número de adaptador asignado por Linux al dispositivo grabador.

Archivo de Canales
Este archivo es leído por la utilidad de sintonizado Tzap, se genera a partir de la salida de la aplicación Scan [44]. Originalmente retorna una línea para cada servicio de todos los canales que se pudieron sintonizar. Un ejemplo de esta salida puede ser:

Archivo 6.3: Salida Scan

```
La Tele HD:557142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 4113:4115:384
La Tele SD:557142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 4097:4099:385
LaTele1SEG:557142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 4129:4131:32920
Monte Carlo HD:563142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 4096:303:32928
Monte Carlo SD:563142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 201:301:32929
Monte Carlo 1 Seg:563142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 544:545:32952
Canal10 HD:575142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 1001:1002:32992
Canal10:575142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 1021:1022:32993
Canal10 1-seg:575142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 1041:1042:33016
TV Ciudad:581142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 513:514:33024
TV Ciudad:581142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 769:770:33025
TV Ciudad:581142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4:FEC_AUTO:QAM_AUTO:
    TRANSMISSION_MODE_AUTO: GUARD_INTERVAL_AUTO:HIERARCHY_NONE: 257:258:33048
```

Como se observa contiene en cada línea el nombre del servicio, la frecuencia y un conjunto de parámetros necesarios para la demodulación de una señal. Al final de cada línea se incluyen también el número de PID del video, el número de PID del audio y el número de identificación del servicio.

Una vez obtenido el archivo anterior, debe ser editado para la utilización con Tzap según nuestras necesidades. Un ejemplo de archivo editado puede ser:

Archivo 6.4: Archivo de Canales

```
La Tele:557142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4: FEC_AUTO:
    QAM_AUTO:TRANSMISSION_MODE_AUTO:GUARD_INTERVAL_AUTO:HIERARCHY_NONE:::
Monte Carlo:563142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4: FEC_AUTO:
    QAM_AUTO:TRANSMISSION_MODE_AUTO:GUARD_INTERVAL_AUTO:HIERARCHY_NONE:::
TNU:569142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4: FEC_AUTO:
```

```
    QAM_AUTO:TRANSMISSION_MODE_AUTO:GUARD_INTERVAL_AUTO:HIERARCHY_NONE:::
Canal 10:575142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4: FEC_AUTO:
    QAM_AUTO:TRANSMISSION_MODE_AUTO:GUARD_INTERVAL_AUTO:HIERARCHY_NONE:::
TV Ciudad:581142857:INVERSION_AUTO:BANDWIDTH_6_MHZ:FEC_3_4: FEC_AUTO:
    QAM_AUTO:TRANSMISSION_MODE_AUTO:GUARD_INTERVAL_AUTO:HIERARCHY_NONE:::
```

Esta edición consiste en dejar una sola línea por cada canal y eliminar los valores correspondientes a programas y servicios, debido a que se necesita grabar el BTS sin demultiplexar programas específicos.

Como nombre de cada línea se indicarán los mismos nombres de canales utilizados en el resto de los archivos de configuración.

6.7.6.2 Manejador de almacenamiento

Debido a que este módulo almacena muestras continuamente, es necesario un mecanismo que administre el volumen almacenado en disco. Para esto se definirá en el archivo de configuración del módulo la cantidad total de espacio destinado a almacenamiento de muestras para todos los canales.

El Módulo Generador se encargará de no superar el espacio asignado borrando muestras antiguas, de esta forma se mantendrá una ventana de tiempo de grabaciones determinada por el espacio asignado.

6.7.7 Módulo Procesador

Este módulo recibirá notificaciones indicando el nombre de una nueva muestra, luego extraerá de ella un conjunto de programas configurados utilizando la aplicación ts2es [61] y para cada programa extraído ejecutará un conjunto de pasos configurados para obtener los valores de los parámetros deseados.

Para esto manejamos el concepto de tarea. Una tarea se identifica por su nombre y se expresa en términos de un comando de terminal y un conjunto de entradas y salidas, las entradas y salidas son los archivos consumidos y generados por la aplicación invocada en el comando de una tarea.

Como entrada a una tarea se puede indicar una salida de otra, con esto logramos definir precedencias entre tareas en una estructura de grafo. Como salida se puede incluir también la salida estándar (StdOut) y la salida de error (StdError).

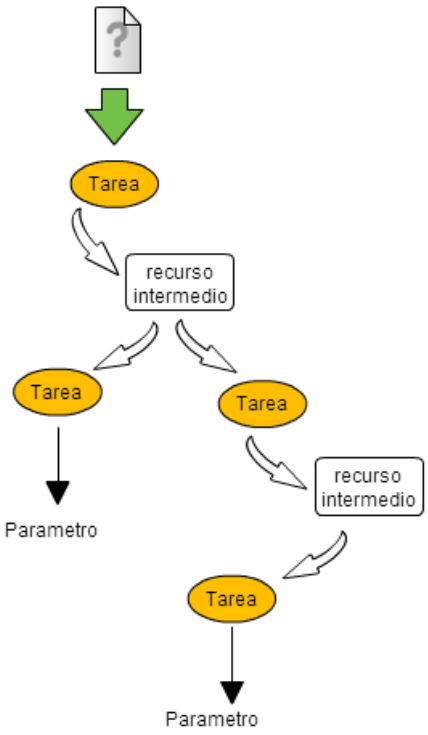

Figura 6.5: Parámetros y recursos intermedios.

De esta forma, los programas elegidos de los canales deseados poseerán un árbol de ejecución que se instanciará en cada ciclo de procesamiento, determinando los nombres de las entradas y salidas en función del nombre de la muestra grabada.

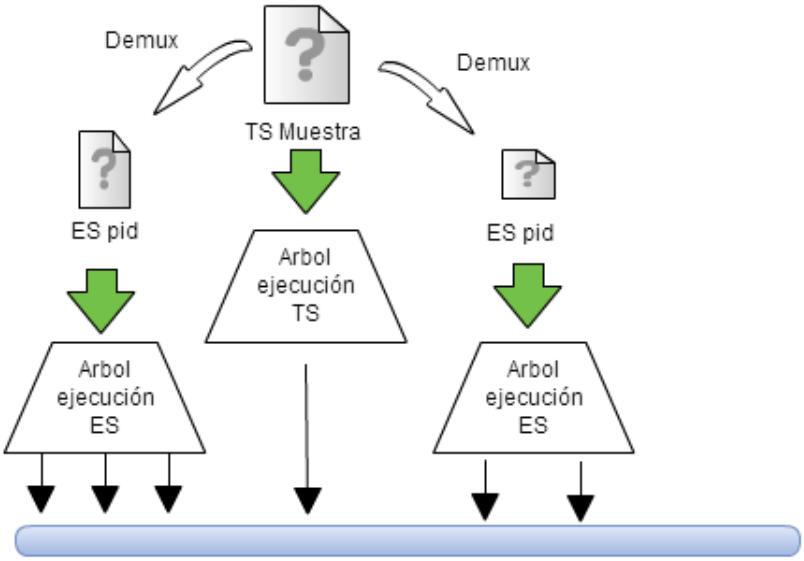

Figura 6.6: Demultiplexación de TS por el sistema.

Las tareas que contienen como entrada únicamente un ES contendrán como entrada el comodín "$ElementaryStreamPid" y las tareas que efectivamente retornen el parámetro buscado (tareas hojas), no declararán salidas y la aplicación invocada en el comando deberá retornar en la salida estándar únicamente el valor del parámetro medido.

Para expresar que parámetros se miden para un programa específico de un canal, se dispondrá de un archivo de parámetros en donde se listará para el programa, las tareas hojas que retornarán los parámetros deseados. Para definir como serán las tareas y precedencias se contará con un archivo de especificación. De modo de admitir procesamiento a partir del archivo de BTSs grabado, se puede definir como entrada de una tarea el comodín "$TSFile" y se incluirá el bloque "TS" en el archivo de parámetros.

Luego a partir de los archivos de parámetros y especificación, este módulo se encargará de ejecutar, a partir de los programas extraídos, las tareas que nos retornarán los parámetros deseados habiendo antes ejecutado todas las tareas precedentes.

6.7.7.1 Configuraciones

A continuación se describirán las configuraciones que se realizarán sobre el Módulo Procesador.

Configuración del módulo
Al igual que el archivo de configuración de módulo del Módulo Generador, consiste en un archivo con configuraciones básicas. Se indican aquí, la ubicación de la carpeta donde están guardadas las muestras, la carpeta que se utilizará para

almacenar los archivos temporales, la carpeta dónde se guardará el log, la ruta al *Named Pipe* en el que se recibe el nombre de la muestra a procesar, la ruta al *Named Pipe* en el que se utilizará para enviar los parámetros al Módulo Recolector, la ruta al archivo de parámetros y la ruta al archivo de especificación que se describirán posteriormente.

Archivo 6.5: Archivo de Configuración del Módulo Procesador

```
<?xml version="1.0" encoding="UTF-8"?>
<conf>
    <recordsPath>
        /home/user/SMTVD/Grabaciones/
    </recordsPath>
    <temporalsDir>
        /home/user/SMTVD/Temporales/
    </temporalsDir>
    <logPath>
        /home/user/SMTVD/LOG/procesador
    </logPath>
    <pipeOutNewRecord>
        /home/user/SMTVD/npNewRecord
    </pipeOutNewRecord>
    <pipeRecolector>
        /home/user/SMTVD/npParameter
    </pipeRecolector>
    <parametersFile>
        /home/user/SMTVD/Conf/ParametrosVideoVQI.xml
    </parametersFile>
    <specificationFile>
        /home/user/SMTVD/Conf/EspecificacionVideoVQI.xml
    </specificationFile>
</conf>
```

Donde:

recordsPath
: indica la ubicación en la que se encuentran guardadas las muestras.

temporalsDir
: indica la ubicación en la que se guardarán los archivos temporales.

logPath
: indica la ubicación donde se guardará el log.

pipeOutNewRecord
: indica la ruta al *Named Pipe* del que se lee el nombre de la muestra a procesar. Cuando se recibe se inicia el procesamiento sobre la muestra.

pipeRecolector
: indica la ruta del *Named Pipe* que se utiliza para enviar los parámetros al Módulo Recolector.

parametersFile
: indica la ruta del archivo de los parámetros que se procesarán para las muestras.

specificationFile
indica la manera en que los parámetros son obtenidos.

Archivo de Parámetros

Archivo 6.6: Archivo de Parámetros Ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<Channels>
    <Channel name="TNU">
        <TS/>
        <PID pid="769">
            <Parameter>slicesPerdidos</Parameter>
        </PID>
        <PID pid="513">
            <Parameter>slicesPerdidos</Parameter>
        </PID>
    </Channel>
</Channels>
```

En este archivo se indica, para cada servicio, que parámetros se medirán para cada programa emitido por el servicio. La forma de obtención de estos parámetros se encuentra descrita en el archivo de especificación.

El elemento“TS" es utilizado en casos de tener que obtener algún parámetro a partir del archivo de BTSs grabado sin pasar por una etapa de demultiplexado, por ejemplo sería útil para contar la cantidad de paquetes TS perdidos. Al incluir un elemento “PID" indicamos al módulo Procesador que demultiplexe del archivo de BTSs, el programa identificado por PID, y que se midan sobre este programa los parámetros indicados.

Archivo de Especificación

Archivo 6.7: Archivo de Especificación Ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<Especificacion>
    <Tarea name="slicesPerdidos">
        <comando>
            perl /home/user/SMTVD/Plugins/Parser.pl \
               "%log%" | /home/user/SMTVD/Plugins/slicer
        </comando>
        <inputs>
            <input name="log">
                LogFFMpeg.StdError
            </input>
        </inputs>
    </Tarea>
    <Tarea name="LogFFMpeg">
```

```
15          <comando>
16              /home/user/SMTVD/Plugins/ffmpeg_g -debug \
17                 pict -i "%ESFile%" -vcodec rawvideo \
18                 -f null /dev/null
19          </comando>
20          <inputs>
21              <input name="ESFile">
22                  $ElementaryStreamPid
23              </input>
24          </inputs>
25          <outputs>
26              <output name="StdError"/>
27          </outputs>
28      </Tarea>
29  </Especificacion>
```

El objetivo de este archivo es indicar la forma en que se obtienen los parámetros listados en Parametros.xml.

Manejamos el concepto de “tarea", una tarea describe los siguientes atributos:

- Están identificadas por un nombre, (se corresponden con nombres de parámetros)
- Un comando de consola con “wildcards", que se sustituirán automáticamente por valores adecuados en cada ejecución. Una serie de inputs, representan recursos necesarios para la ejecución del comando (p.e.; archivos intermedios).
- Una serie de outputs, representan recursos que produce un comando.

Para indicar que una tarea depende de otra, es decir que necesita como entrada un recurso producido por otra tarea, se indica como input [nombreTarea.nombreSalida] de la tarea precedente. En cambio para indicar que la tarea se ejecuta a partir del archivo de BTSs grabado o de uno de sus ES demultiplexados, se indica “$TSFile" o “$ElementaryStreamPid".

Para cada tarea se indica como *output* todo recurso que produce, cada recurso producido se escribe en un archivo para que sea leído por otra tarea.

Los comandos utilizados producen recursos de dos formas, declarando en su invocación (comando que se escribe en la terminal) archivos de salida, o escribiendo el el StdOut y StdError. Para contemplar esta situación, se indica como nombre del output de la tarea “StdOut" y “StdError".

Las tareas finales, que no poseen sucesoras y computan efectivamente el valor de un parámetro, deben devolver su salida en el StdOut.

Con esta sintaxis logramos armar, para cada archivo de BTSs grabado y ES extraído, un árbol de comandos para obtener los valores de los parámetros configurados.

6.7.7.2 Protección de sobrecarga

Al recibir una notificación de nueva muestra, se ejecutan los árboles de ejecución que determinan los parámetros. Debido a que la estructura que se utiliza para la obtención de los mismos es leída al inicio de la ejecución del módulo y se mantiene sin modificaciones, es siempre igual y por lo tanto sería deseable que se reutilice en cada ejecución.

Esta política fue la adoptada en el presente proyecto. De esta manera, si mientras un grafo se ejecuta aparece una nueva muestra, esta intentará ejecutar sobre el mismo grafo produciendo que los datos se sobrescriban y generando errores en los parámetros a obtener. Esta situación puede darse si no se poseen recursos de hardware suficientes para obtener los parámetros configurados antes que la siguiente muestra se obtenga, es decir, el tiempo insumido para la obtención de parámetros excede el tiempo de duración de la muestra. Para proteger al sistema de esta situación es necesario definir un método de protección.

El sistema de protección de sobrecarga, funciona de la siguiente manera: Al aparecer una nueva muestra, se consulta si los nodos raíces están en ejecución. En caso de estarlos, la ejecución de ese grafo se descarta, y en caso contrario se inicia. Si durante la ejecución del grafo, se llega a que un nodo sucesor está en ejecución, se espera a que termine y una vez finalizado se continúa con el proceso.

Bajo esta estrategia cada grafo de ejecución posee un comportamiento similar a un *buffer*, en donde se controla que la tasa de arribos (las nuevas muestras a analizar) no supere la tasa de salidas (la obtención de todos los parámetros de cada muestra).

Además de evitar el problema mencionado, esta estrategia habilita a que los grafos ejecuten haciendo *Pipelining* con las muestras. El *Pipelining* se da en caso de que la raíz haya culminado y que el tiempo entre éstas no alcance para finalizar la obtención de los parámetros.

Es importante aclarar que si una tarea se encuentra esperando a que alguno de sus sucesores termine se considera como si estuviera en ejecución.

Por ejemplo, partamos del árbol de ejecución de la 6.7, donde se está ejecutando la última tarea sobre una muestra.

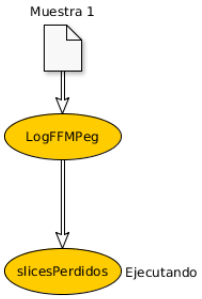

Figura 6.7: Ejemplo de protección de sobrecarga 1.

Al llegar una nueva muestra, se iniciará su ejecución debido a que la raíz no está ejecutando.

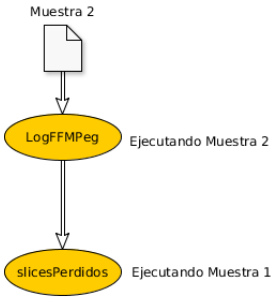

Figura 6.8: Ejemplo de protección de sobrecarga 2.

Luego, si la tarea “LogFFMpeg" culminara antes de que la tarea “slicesPerdidos" termine su ejecución de la primer muestra, se obtendría el siguiente resultado.

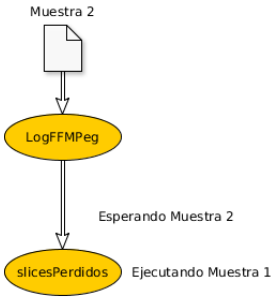

Figura 6.9: Ejemplo de protección de sobrecarga 3.

Al terminar la tarea “slicesPerdidos" sobre la primer muestra, comenzaría a ejecutar dicha tarea sobre la segunda.

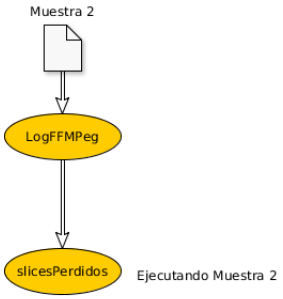

Figura 6.10: Ejemplo de protección de sobrecarga 4.

Ahora bien, si llegara una muestra 3 ocurriría lo mismo que explicamos anteriormente.

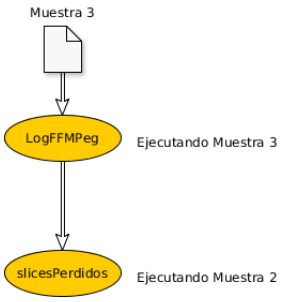

Figura 6.11: Ejemplo de protección de sobrecarga 5.

Pero, si la ejecución de la tarea “LogFFMpeg" toma demasiado tiempo, podría suceder que llegara una nueva muestra. Esta muestra sería descartada ya que la raíz del árbol se encuentra en ejecución.[2]

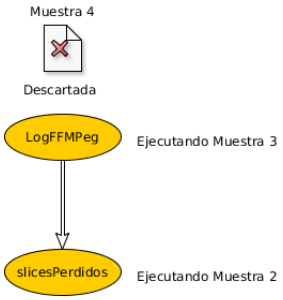

Figura 6.12: Ejemplo de protección de sobrecarga 6.

Cabe aclarar que si durante el estado de la figura 6.9 hubiera llegado una nueva muestra, ocurriría lo mismo que acabamos de mencionar, debido a que

[2]Nota: La ejecución de la tarea “slicesPerdidos" podría haber terminado al llegar la cuarta muestra.

el “LogFFMpeg" se encuentra esperando por “slicesPerdidos" y por tanto se considera en ejecución.

6.7.7.3 Manejador de almacenamiento

Debido a que este módulo genera archivos intermedios usados en la obtención de parámetros, es necesario también un mecanismo que los elimine automáticamente cuando ya no se los necesite.

Los archivos intermedios están determinados como salidas de tareas, y serán eliminados una vez obtenidos todos los parámetros correspondientes al grafo de ejecución para cada muestra.

6.7.8 Obtención de potencia de señal

Si bien el sintonizador utilizado reporta la potencia de señal, lo hace utilizando un valor que no es dado en ninguna unidad de potencia útil. Este valor se utiliza para reportar la potencia de señal como un porcentaje realizando la operación

$$P = \frac{vr \times 100}{mv} \tag{6.1}$$

En donde:
vr : es el valor reportado por el dispositivo
mv : es el máximo valor medible con el dispositivo

Este valor depende del dispositivo que se esté utilizando pero en algunos usos como por ejemplo la aplicación tzap utilizada en este proyecto, se utiliza un máximo de 0xFFFF [62].

Observando el código fuente de un *driver* para dispositivos de tv digital (módulo dib8000 del *kernel* del Linux [63]), se halló que es posible encontrar un valor aproximado de la potencia de señal en dBm mediante la creación de manera experimental de una tabla de valores.

Para crear la tabla se deberá medir el valor reportado por el dispositivo mientras se envía señal de potencia conocida mediante equipamiento de laboratorio y se anota el valor medido y su correspondiente señal. Este procedimiento se lleva a cabo para un amplio rango de valores posibles de señal y con intervalos regulares. Luego para determinar el valor de señal aproximado para un valor cualquiera, se determinan los dos puntos de la tabla más cercanos a el y se interpola mediante una recta que une los tres puntos usando la fórmula de interpolación lineal por dos puntos conocidos

$$y = y_0 + (y_1 - y_0)\,\frac{x - x_0}{x_1 - x_0} \tag{6.2}$$

Si el punto a evaluar se ubica por fuera del rango establecido, en este caso se opta por devolver el valor máximo o mínimo de la tabla al igual que en el código visto en el *driver* dib8000 [63].

6.7.9 Módulo Recolector

El rol de este módulo es generar las notificaciones a partir de los valores de los parámetros medidos y enviarlas al agente RNR para que este lo envía al Sistema Centralizado.

6.7.9.1 Configuraciones

Configuración del módulo
Consiste en un archivo XML en donde se indica el nombre que tomará la estación, el pipe de donde leerá los valores de los parámetros y dirección IP y puerto del agente RNR local.

Archivo 6.8: Archivo de Configuración del Módulo Recolector

```
<?xml version="1.0" encoding="UTF-8"?>
<conf>
    <pipeRecolector>
        /home/user/SMTVD/npParameter
    </pipeRecolector>
    <NombreEstacion>ChrBox ASUS</NombreEstacion>
    <RNRHost>192.168.1.43</RNRHost>
    <RNRPort>8182</RNRPort>
</conf>
```

Donde:

pipeRecolector
: indica la ruta del Named Pipe del que se lee los parámetros a enviar al Sistema Centralizado.

NombreEstacion
: indica el nombre de la estación. Se utiliza para identificar las estaciones.

RNRHost
: indica la IP del host RNR (Sistema Centralizado).

RNRPort
: indica el puerto en el que el host RNR espera los mensajes.

6.7.10 Sistema Centralizado

Los objetivos del Sistema Centralizado son, por un lado, almacenar y exhibir los valores obtenidos en las mediciones de parámetros de calidad en las estaciones, junto con los valores obtenidos luego de computar los modelos de calidad sobre estos.

Por otro lado, en base a los datos recolectados, deberá reconocer situaciones anormales o de calidad deficiente y se obtendrán las muestras correspondientes como evidencia y para posterior análisis.

Como se menciona anteriormente, el transporte de los valores de parámetros medidos en las estaciones hacia el Sistema Centralizado se realiza utilizando el

protocolo RNR. Para esto el equipo en donde ejecutará este sistema deberá ejecutar un agente RNR.

Por simplicidad de la solución, el Sistema Centralizado se descompone en dos módulos, un Endpoint de Notificaciones y una Interfaz para la exhibición de la información obtenida. Estos módulos constituyen dos ejecutables independientes que se comunican únicamente mediante el almacenamiento, es decir, no existe una interfaz de comunicación directa entre ambos ejecutables.

6.7.10.1 Almacenamiento de la información

El almacenamiento de la información se realizará en una base de datos no relacional Mongo DB [59], sobre esta base se insertarán objetos que representan las notificaciones recibidas de cada una de las estaciones y objetos que representan los valores obtenidos de computar las funciones de calidad.

Se deberá disponer también un directorio para el almacenamiento de muestras obtenidas, a modo de evidencia, desde las estaciones.

6.7.10.2 Endpoint de recepción de notificaciones

Este módulo cumple con las responsabilidades de almacenar la información recibida en las notificaciones y a partir de esto, la información obtenida computando las funciones de calidad. Para esto, este módulo se comunicará con el Agente RNR de modo de obtener las notificaciones correspondientes.

Una vez obtenida una notificación, se almacenará en la base de datos un objeto que la representa y se verifica si se está en condiciones de computar una función de calidad a partir de los datos almacenados. Es decir, se revisa para cada modelo de calidad, si se han recibido todos los parámetros necesarios correspondientes a cada programa, canal, ubicación y tiempo, en este caso se computa la función correspondiente y se almacena en la base de datos un nuevo objeto representando el valor de calidad obtenido.

Será este módulo quien implemente la lógica necesaria para la detección de situaciones anormales y obtención de evidencias.

En el presente proyecto se realizó un prototipo de este módulo, en donde se computan solamente las funciones del modelo de calidad candidato, debido a que se trata de un prototipo, en el presente trabajo las funciones de calidad no son configurables.

Siguiendo la misma línea que los otros módulos, el *Endpoint* de Notificaciones se ha implementado en Java[49].

6.7.10.3 Front-end Web

Para mostrar a los usuarios la información obtenida a partir de los datos recibidos por las estaciones, se diseñó un *Front-End* web que cuenta con un panel lateral en donde se puede seleccionar una estación y un panel central donde se exhiben gráficas con los valores de los parámetros medidos y los resultados de aplicar los

modelos de calidad.

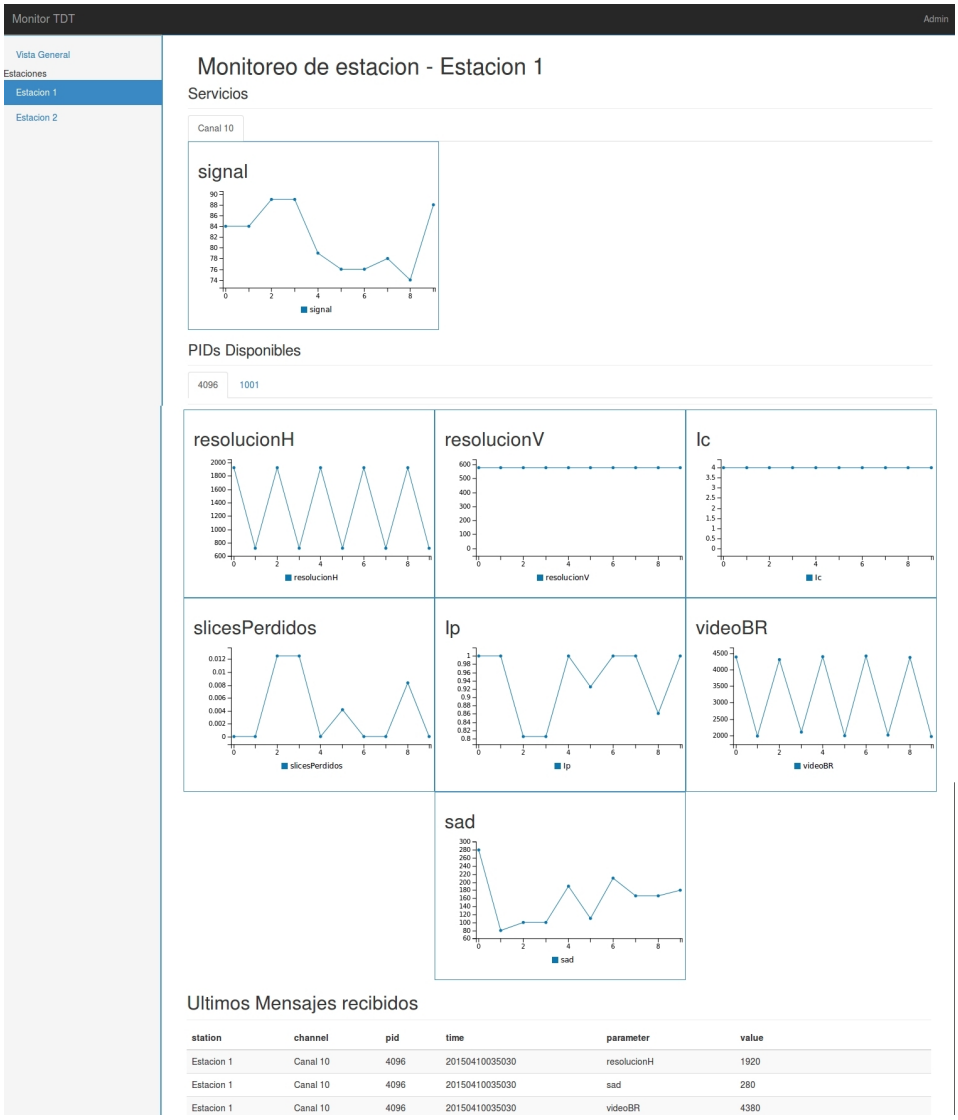

Figura 6.13: Diseño Front-End Web.

La arquitectura de la aplicación web consta de un solo servidor central donde se encuentra el servidor de la aplicación, el servidor de bases de datos y el Endpoint donde se reciben los datos de todos los nodos del sistema de monitoreo. De ser necesario por motivos de escala del sistema, se podrían separar estos tres sistemas en equipos separados sin inconvenientes.

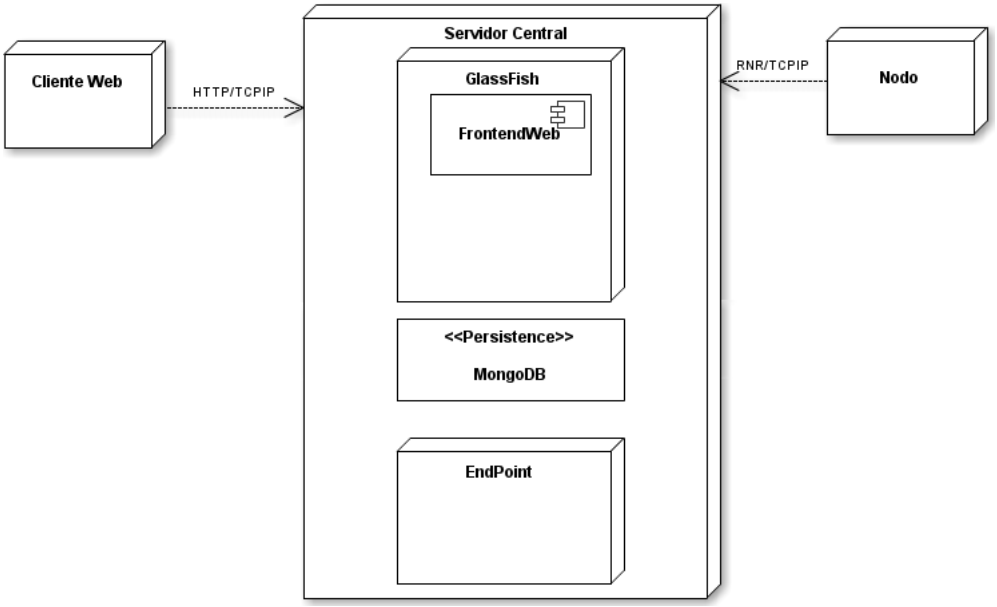

Figura 6.14: Diseño Front-End Web.

Debido a la poca cantidad de vistas con las que cuenta el sistema y que la mayoría de las llamadas posibles a la aplicación web están relacionadas entre sí, se optó por un patrón de diseño fachada que provee un sólo punto de acceso de las llamadas de los clientes web al sistema.

La aplicación fue desarrollada en Java utilizando el *framework* Spring MVC y Spring Data para el acceso a la base de datos. Corre en un servidor Glassfish y los datos son consultados de una base de datos MongoDB donde son almacenados por el *Endpoint*. El único medio de comunicación entre el Endpoint y la aplicación web es a través de datos en la base de datos.

La vista general de la aplicación, muestra una lista con los identificadores de todos los nodos de los que se han recibido datos y el estado de los mismos, para tener una idea general del estado del sistema. Dentro de esta vista general, se puede acceder a una vista detallada al seleccionar cualquier estación lo que mostrará en más profundidad los datos que han llegado y se podrán ver gráficas de los diferentes parámetros medidos.

Al ser una aplicación para monitoreo de datos, se intentó evitar que el usuario deba refrescar la página para actualizar los datos por lo que todos los datos se actualizan periódicamente utilizando funciones JavaScript que realizan llamadas al servidor intercambiando mensajes JSON con el éste.

Para la presentación de la aplicación se utilizaron los *frameworks* JQuery y Bootstrap y para las gráficas se utilizó la librería C3.js.

Como los parámetros medidos por los nodos dependen del modelo de calidad utilizado y éste además puede cambiarse en cualquier momento, se optó por hacer

que la aplicación web no haga referencias estáticas a los parámetros del modelo de calidad, lo que evitaría tener que modificarla cada vez que se agrega un nuevo parámetro al modelo. Para lograr esto definimos una notificación como un hash de propiedades de manera de poder mostrar los nombres de los parámetros y sus valores sin necesidad de conocer los nombres de antemano.

Acceso a datos
Para aprovechar las facilidades que provee SpringData para MongoDB, como son las consultas que se pueden realizar solamente declarando funciones en una interfaz sin necesidad de implementarlas, sin perder la posibilidad de utilizar consultas personalizadas se modificó la estructura sugerida por la documentación de SpringData de manera de que se permita introducir nuevas funciones sin perder la autoconfiguración.

Para ello se declara una clase abstracta como normalmente se hace en estos casos a la que llamamos MessageRepository que extienda la clase MongoRepository. Si sólo se quisieran utilizar las consultas autoimplementadas por el *framework* sólo sería necesario utilizar declarar los nombres de los métodos. Como la intención es poder además utilizar métodos implementados personalizadamente, definimos una interfaz que declare los métodos que necesitamos declarar y hacemos que la clase MessageRepository realice esta interfaz. Por último declaramos una clase que implemente la interfaz con los métodos manuales siguiendo la convención de nombres de SpringData para los nombres de las clases. De esta forma Spring utilizará para los métodos definidos como automáticos la implementación autogenerada a partir del nombre del mismo y para los métodos definidos dentro de la interfaz Custom las implementaciones manuales definidas en la clase Impl.

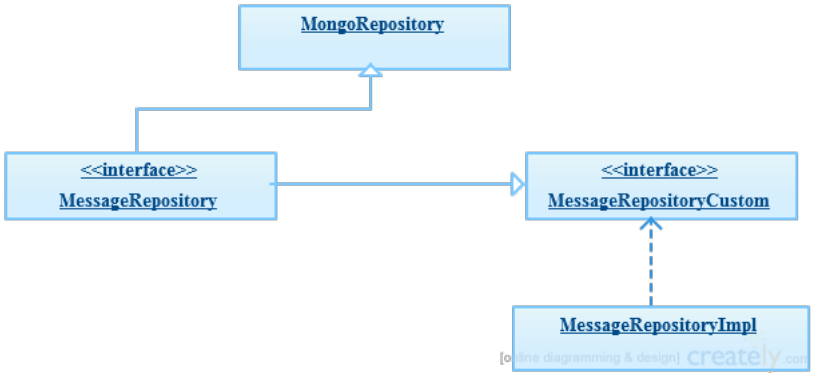

Figura 6.15: Acceso a datos Front-End Web.

Capítulo 7

Hardware y software utilizado

En este capítulo se describirá el hardware y software utilizado a lo largo de este proyecto, indicando el sistema base, los sintonizadores y finalmente el hardware de las estaciones. Además se explicará la razón de tal elección.

7.1 Sistema Base

Con el objetivo de independizar el software construido para las estaciones del hardware subyacente, se decidió implementar el sistema utilizando la plataforma Java [49], esta decisión fue respaldada por el equipo de trabajo debido a la familiaridad con el lenguaje.

Para las estaciones se decidió utilizar sistemas operativos Linux [45] con el objetivo de potenciar la utilización del sistema construido, ya que el utilizar sistemas *open source* nos permite adaptarlos de acuerdo a las necesidades específicas de cada realidad.

En este proyecto se utilizó la distribución Ubuntu [46], debido a que está disponible una compilación adecuada al hardware elegido. Luego, con el objetivo de disminuir la utilización del uso de recursos por ejecutables innecesarios para el proyecto, fueron deshabilitados los servicios relativos a interfaces gráficas, audio, actualizaciones, reportes de fallas, servicios de impresoras, *bluetooth*, entre otros.

La administración y configuración de las estaciones se realizó vía SSH, para esto se debió instalar la suite OpenSSH [47], ya que no es incluida en forma nativa en la distribución elegida.

Para la obtención de algunos parámetros se utilizaron *scripts* en Perl [48], debido a su gran potencia para el procesamiento de textos.

7.2 Sintonizadores digitales

En primer lugar investigamos que sintonizadores digitales estaban disponibles en plaza y que interfaces proveen para la interacción con estos desde una PC y de-

terminamos como mejor alternativa la utilización de Dongles USB.

Según el listado de dispositivos ISDB-T USB de [50] para los cuales se proveen *drivers* para Linux, consideramos como mejor alternativa el Dongle Geniatech MyGica S870 Full Segment [51], que fue importado de China a un costo de USD 28.

La decisión se basó en que fue el único dispositivo listado en [52] disponible en plaza y además provee también *drivers* para Windows, aportando al sistema mayor flexibilidad e independencia.

Está basado en el *chipset* Dibcom dib-0800, generalmente soportado en Linux y el *kernel* ya incluye varios *drivers* para chips de su fabricante (Parrot OEM) [13].

Vale aclarar que a pesar de que el *kernel* ofrece soporte para estos chips, ISDB-Tb no está completamente soportado por el *kernel* de Linux por lo que los sintonizadores conectados son vistos por el sistema como sintonizadores DVB-T, lo que para los objetivos de este proyecto no representa un problema ya que se trabaja a nivel de los contenidos elementales emitidos.

7.3 Estaciones

Debido a la elección de una arquitectura distribuida en donde se toman mediciones de parámetros en estaciones de procesamiento remoto, se debió elegir equipos adecuados para esta tarea.

La potencia de hardware necesaria está determinada por las tareas necesarias para la obtención de parámetros y por la cantidad de canales y programas a monitorizar. Basándonos en el modelo candidato investigamos sobre dispositivos disponibles en plaza con el objetivo de encontrar, para las estaciones, equipos suficientemente potentes pero manteniendo los costos acotados.

Las pruebas realizadas consistieron en la ejecución de las aplicaciones necesarias para la extracción de parámetros de una muestra de video de una emisión de una canal de televisión. Debido a que cada ciclo de grabación y procesado está determinado por las muestras tomadas, tomamos como criterio de aceptación que el tiempo insumido para la obtención de los parámetros necesarios no supere el tiempo de duración de la muestra.

En primer lugar experimentamos con una notebook Acer Extensa 5620 [53] con las siguientes características:

CPU	Intel Core 2 Duo T5250 1.5 GHz
Memoria	4 GB DDR2
Almacenamiento	Mecánico

También realizamos pruebas sobre un RaspBerry Pi modelo B [54], con las siguientes características:

CPU	ARM 1176JZ-F 700 MHz
Memoria	512 MB SDRAM
Almacenamiento	Tarjeta SD

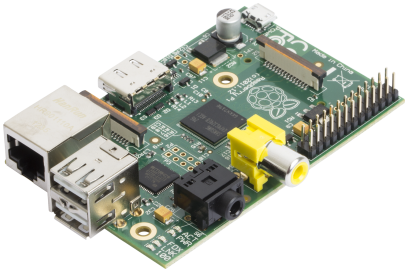

Figura 7.1: RaspBerry Pi Model B.

Este dispositivo fue rápidamente descartado debido a que el tiempo necesario para la obtención de parámetros fue de aproximadamente 4 veces el tiempo de duración de una muestra. Luego experimentamos con la laptop del Plan Ceibal [55] Positivo BGH CL11 [56] con las siguientes especificaciones:

CPU	Intel Celeron Dual Core 1017U 1.6 GHz
Memoria	1 GB DDR3
Almacenamiento	SSD

Tanto las pruebas en la laptop Acer como en la Positivo BGH fueron satisfactorias, y sus prestaciones de hardware fueron tomadas como referencia para la potencia necesaria para las estaciones. Investigando en plaza dispositivos SoC y Barebones consideramos como opciones los siguientes dispositivos:

Intel NUC DN2820FYKH

CPU	Intel Celeron N2820 Dual Core 2.13 GHz
Memoria	No incluido
Almacenamiento	No incluido

Figura 7.2: Intel NUC DN2820FYKH.

Zotac ZBOX-BI320-U

CPU	Intel Celeron 2957U Dual Core 1.4 GHz
Memoria	No incluido
Almacenamiento	No incluido

Figura 7.3: Zotac ZBOX-BI320-U.

Asus Chromebox M004U

CPU	Intel Celeron 2955U Dual Core 1.4 GHz
Memoria	2 GB DDR3L
Almacenamiento	SSD 16GB

Figura 7.4: Asus Chromebox M004U.

HP Chromebox CB1-014

CPU	Intel Celeron 2955U Dual Core 1.4 GHz
Memoria	2 GB DDR3L
Almacenamiento	SSD 16GB

Figura 7.5: HP Chromebox CB1-014.

Se optó por las Chromebox ASUS y HP debido a su bajo costo y que incluyen memoria RAM, almacenamiento de 16 GB y la posibilidad de utilizar tanto conexiones WiFi como Ethernet para la transferencia de datos. Además provee 4 puertos USB 3.0 para la conexión de los sintonizadores digitales.

Cabe destacar que la Chromebox pose como sistema operativo ChromeOS [57], pero para cumplir con el sistema a desarrollar se debió instalar Linux.

También se investigó sobre dispositivos basados en ARM, en particular los que son conocidos como Android TV. Uno de estos dispositivos es conocido como CS918 [58] y posee las siguientes especificaciones:

CPU	ARM Cortex-A9 Quad Core 1.8 GHz
Memoria	2 GB DDR3
Almacenamiento	Tarjeta SD

Figura 7.6: CS918.

Estos dispositivos son considerablemente más potentes y menos costosos que los anteriores pero no fueron tomados en cuenta como candidatos, debido a la no disponibilidad de *drivers* para los sintonizadores.

Capítulo 8

Resultados Obtenidos

En este capítulo se presentan las pruebas realizas así como los resultados obtenidos a partir de las mismas. El objetivo principal será evaluar tanto el comportamiento del sistema como el hardware seleccionado.

8.1 Descripción

Para validar el software construido y la arquitectura propuesta, se configuró el sistema de modo de computar el modelo candidato sobre programas de un conjunto de canales de televisión. El objetivo de esto es verificar que efectivamente los módulos Generador y Procesador trabajen según una configuración y sean medidos y registrados la totalidad de los parámetros configurados.

Se realizaron pruebas sobre el hardware elegido para el prototipo con el objetivo de evaluar qué posibilidades de procesamiento se posee en función de la potencia de las estaciones. Sobre las mismas se ejecutaron los módulos Generador, Procesador y Recolector, manteniendo en un nodo aparte el Servidor Central.

Utilizando el modelo candidato, las pruebas se llevaron a cabo variando las configuraciones de los módulos en términos de cantidad de canales y sus servicios a monitorizar. En cada prueba se registró con la aplicación NMON [60] la utilización de CPU.

Como se detalla anteriormente, el mecanismo de protección de sobrecarga descarta ejecuciones de procesamientos de muestras al alcanzar la saturación del nodo. Es por este motivo que una prueba se considera satisfactoria si se procesan el 100% de las muestras tomadas, es decir, que efectivamente se logran obtener todos los parámetros configurados para cada una de las muestras obtenidas de los canales y programas a monitorizar en la estación. En caso de alcanzarse la saturación del nodo, se verificará que el mecanismo de protección de sobrecarga efectivamente se comporte de manera descrita.

De esta forma se evalúa la capacidad del hardware elegido y se verifica la correctitud de la plataforma construida.

8.2 Ambiente de pruebas

8.2.1 Preparación de la estación

Como se mencionó anteriormente, para evitar consumo de recursos innecesarios, se deshabilitaron un conjunto de servicios. A continuación se indican cuáles fueron y a qué funcionalidad corresponden.

Servicio	**Descripción**
alsa-utils	Advanced Linux Sound Architecture, audio
avahi-daemon	Avahi DNS service discovery
cgmanager	Daemon to manage cgroups
cups	Common Unix Printing System
cups-browsed	cups usando Avahi
ondemand	CPU scaling daemon
pulseaudio	PulseAudio sound server
lightdm	LightDM desktop display manager
whoopsie	Ubuntu Error Reporting daemon
bluetooth	Bluetooth service
plymouth	Bootsplash animation during booting
apport	Internal error reporting

8.2.2 Canal de televisión de prueba

Para llevar a cabo las pruebas utilizamos equipamiento del Laboratorio de Televisión Digital del LATU [83]. Este equipamiento nos permite emitir un canal de televisión de prueba en base a videos. Los videos utilizados en este canal son los que dispone el LATU para la homologación de equipos de televisión en general.

Para estas pruebas, se configuró un canal de televisión compuesto por dos servicios, uno HD y otro SD, ambos con audio. Las características de los programas de video son las siguientes:

	SD	**HD**
Bitrate promedio	4 Mbps	10 Mbps
Resolución	720×576	1920×1080
Tipo de escaneo	Entrelazado	Entrelazado

8.3 Pruebas realizadas

Las pruebas consistieron en realizar el proceso de grabación y procesamiento sobre dos minutos de los videos anteriormente mencionados y notificación de los parámetros obtenidos a partir de los mismos, cuatro veces consecutivas. De esta forma se evalúa la capacidad del hardware seleccionado y la consistencia del sistema construido.

Debido a que son dos minutos de video, al ser fraccionados en muestras de 10 segundos, se obtienen 12 muestras.

El criterio de aceptación de que dicho hardware sea útil como estación remota para el cálculo de las componentes de calidad, será que sea posible ejecutar el 100% de las muestras, es decir, que el mecanismo de protección no descarte procesamiento.

Se realizaron pruebas para la componente de calidad de video afectada por la propagación y para la de calidad de video de origen.

8.3.1 Pruebas computando la componente de calidad afectada por propagación

Para estas pruebas, se configuró en la estación para computar los *slices* afectados por las pérdidas de datos durante la propagación de la señal.

Debido a los procesamientos se inician a partir de las muestras obtenidas a intervalos periódicos, observamos que la utilización de CPU presenta ráfagas de alta y baja utilización. Esto se debe al funcionamiento del Módulo Procesador, en donde los ciclos de procesamiento son iniciados luego de grabadas las muestras y esto se hace en intervalos periódicos.

Para estas pruebas se configuró el módulo procesador de modo de obtener el porcentaje de *slices* de video perdidos y además se inició la aplicación FEMonReader para registrar la potencia de señal.

La siguiente tabla resume las pruebas y sus resultados para la **Chromebox ASUS M004U**. [1]

Prueba	**Programas**	**# Parámetros**	**Resultado**
1	SD	12/12	Satisfactorio
2	HD	12/12	Satisfactorio
3	SD,HD	24/24	Satisfactorio
4	SD,SD	24/24	Satisfactorio
5	HD,HD	17/24	No satisfactorio

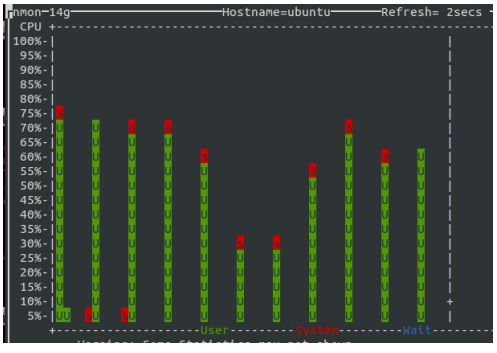

Figura 8.1: Prueba 1.

[1] Las pruebas 4 y 5 se realizaron procesando simultáneamente la misma muestra.

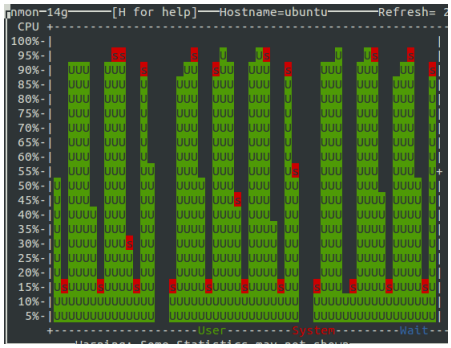

Figura 8.2: Prueba 2.

Figura 8.3: Prueba 3.

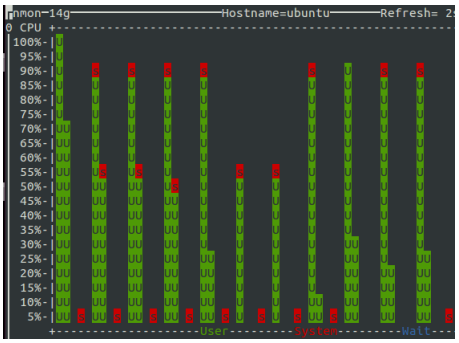

Figura 8.4: Prueba 4.

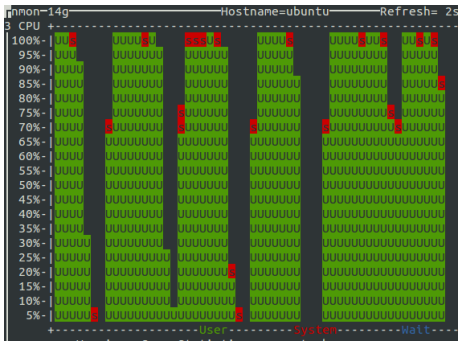

Figura 8.5: Prueba 5.

8.3.2 Pruebas computando la componente de calidad de video en origen

Para estas pruebas se configuró el módulo procesador de modo de obtener los parámetros resolución, *bitrate* y SAD, que son los correspondientes a la calidad de video de origen.

La siguiente tabla resume las pruebas y sus resultados para la **Chromebox ASUS M004U**.

Prueba	**Programas**	**# Parámetros**	**Resultado**
6	SD	44/48	No satisfactorio
7	HD	43/48	No satisfactorio

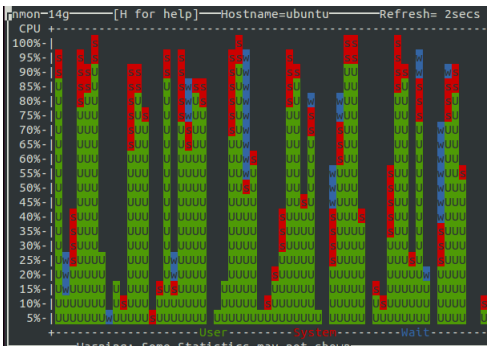

Figura 8.6: Prueba 6.

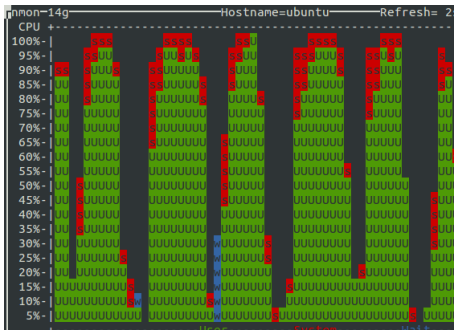

Figura 8.7: Prueba 7.

8.4 Pruebas realizadas fuera del hardware candidato

Para evaluar el desempaño del sistema en hardware de mayor potencia, se realizaron pruebas en una laptop **Toshiba Satelite S55 A5279** con la siguientes características:

CPU	Intel I7 4700MQ 2.4 GHz
Memoria	16 GB DDR3
Almacenamiento	Intel 530 SSD 240GB

8.4.1 Prueba computando la componente de calidad afectada por propagación

Prueba	**Programas**	**# Parámetros**	**Resultado**
8	SD,HD,SD,HD	48/48	Satisfactorio

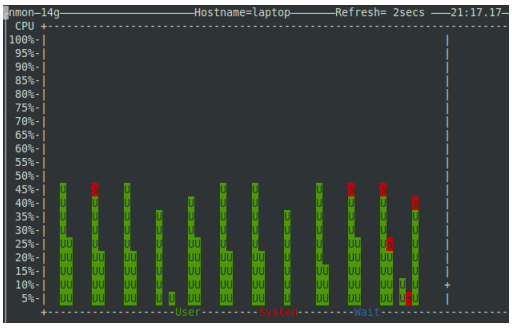

Figura 8.8: Prueba 8.

8.4.2 Pruebas computando la componente de calidad de video en origen

Prueba	Programas	# Parámetros	Resultado
9	SD	48/48	Satisfactorio
10	HD	48/48	Satisfactorio
11	SD,HD	96/96	Satisfactorio
12	SD,SD	96/96	Satisfactorio
13	HD,HD	96/96	Satisfactorio
14	SD,HD,SD,HD	184/192	No satisfactorio

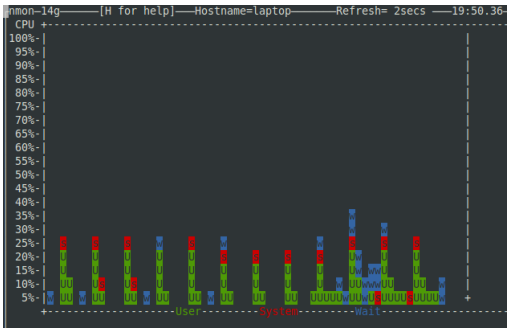

Figura 8.9: Prueba 9.

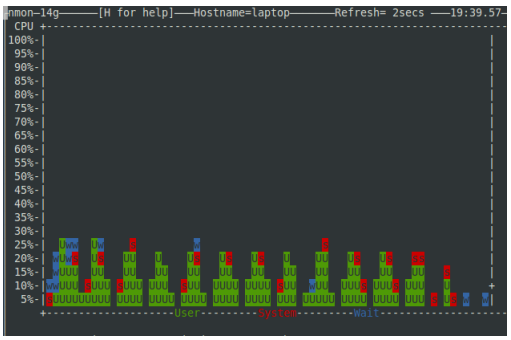

Figura 8.10: Prueba 10.

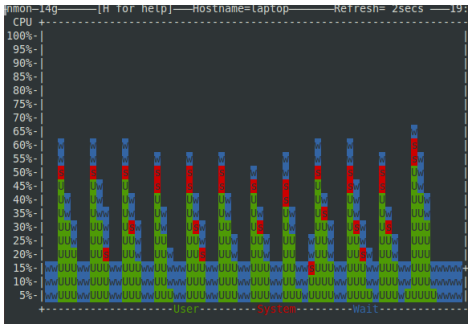

Figura 8.11: Prueba 11.

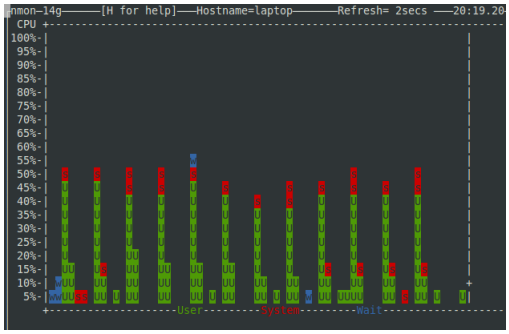

Figura 8.12: Prueba 12.

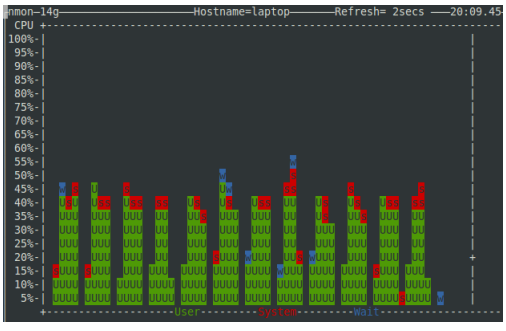

Figura 8.13: Prueba 13.

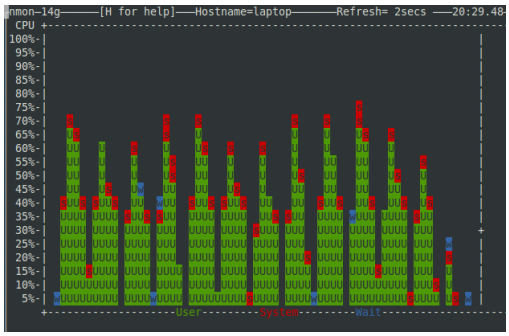

Figura 8.14: Prueba 14.

8.5 Conclusiones sobre los resultados obtenidos

Según la configuración correspondiente al cómputo de la componente de calidad afectada por propagación, se logró procesar en el hardware elegido, la totalidad de las muestras tomadas procesando los programas SD y HD del canal de prueba emitido. Es decir, por cada ubicación geográfica donde se desee monitorizar la calidad afectada por propagación de los programas SD y HD, sera suficiente contar con una unidad de este dispositivo por cada canal emitido.

La potencia del hardware elegido no resultó suficiente para el cómputo de la componente de calidad de video en origen. Para esta tarea se debe recurrir a equipos mas potentes, como el citado en sección anterior, capaz de procesar hasta dos programas en simultáneo.

Capítulo 9

Conclusiones y trabajo a futuro

En este capítulo se comentan los resultados más importantes y la contribución que este trabajo aporta. Se incluye también el trabajo a futuro que comenta las posibles extensiones del proyecto para lograr un sistema más completo.

9.1 Conclusiones

En el presente proyecto se logró diseñar y construir un sistema capaz de dar soporte a la obtención de valores de calidad proveyéndole al mismo la forma de obtenerlos, es decir, no se trata de una implementación específica de un modelo, sino de una generalización de los mismos a través de sus parámetros. Gracias a la arquitectura del sistema, los nodos pueden ser ubicados en distintas zonas geográficas y así tomar medidas de forma distribuida. También el sistema consta de un histórico de las muestras procesadas cuya capacidad también es configurable. A través de estas funcionalidades que provee el sistema, se logra el cumplimiento de los objetivos primarios.

Por otro lado, para cumplir con los objetivos secundarios, se construyó el sistema de forma tal que admita la configuración de los canales y servicios a ser monitorizados. Además se construyó un prototipo de una aplicación que toma medidas de la potencia de señal recibidas por el sintonizador y las envía en forma de porcentaje. De esta forma, se podría extender el sistema realizando aplicaciones como la mencionada que tomen medidas de otros parámetros de utilidad.

Gracias a la arquitectura elegida, consideramos haber alcanzado los objetivos de flexibilidad propuestos manteniendo la simplicidad del diseño de la solución.

Para la verificación del sistema se ha seleccionado un modelo de calidad de video y se ha experimentado con hardware comprado en el exterior obteniendo resultados satisfactorios.

9.2 Trabajo a futuro

En el sistema construido se omitieron aspectos relativos a la seguridad del sistema. En lo que respecta a la transmisión de información sobre la red no se utilizaron mecanismos que garanticen integridad y confidencialidad. Tampoco se ha tenido en cuenta la creación de un subsistema de usuarios y permisos.

El mecanismo utilizado por el grupo de trabajo para la administración y configuración de las estaciones remotas consiste en el acceso remoto mediante SSH [67]. Sería deseable en una próxima iteración diseñar un mecanismo en donde se pueda, a través de una interfaz gráfica, editar la configuración de las estaciones, habilitando o deshabilitando parámetros a medir o la forma en que éstos se obtienen.

Se experimentó con un modelo de calidad de video y se verificó su funcionamiento en el hardware propuesto. Debido a que el sistema construido soporta también modelos de audio e interactividad, se deberá experimentar con ellos y determinar hardware adecuado para estos casos.

9.3 Proyecto SMTVD

Actualmente se lleva a cabo el proyecto SMTVD impulsado por ANII[84] y DINATEL[85], que plantea una serie de objetivos en donde algunos de ellos son compartidos con este proyecto. De esta manera se integrarán, reutilizando gran parte de los objetos desarrollados y siguiendo los lineamientos de arquitectura definidos. En particular, se reutilizan los módulos correspondientes a las estaciones remotas y se plantean una serie de mejoras sobre el servidor central, además de la inclusión del soporte de modelos con referencia.

Referencias

[1] Wikipedia, *Televisión Digital*.
http://es.wikipedia.org/wiki/Televisi%C3%B3n_digital
Consulta Julio 2014.

[2] Wikipedia, *Apagón Analógico*.
http://es.wikipedia.org/wiki/Apag%C3%B3n_anal%C3%B3gico
Consulta Enero 2015.

[3] Televisión Digital Abierta, *¿Qué es la Televisión Digital Abierta?*.
http://www.tvd.gub.uy/queeslatvd_definicion.php
Consulta Enero 2015.

[4] Wikipedia, *Redes de Sensores*.
http://es.wikipedia.org/wiki/Red_de_sensores
Consulta Febrero 2015.

[5] Universitat de València, *Redes de Sensores*.
http://www.uv.es/~montanan/ampliacion/trabajos/Redes%20de%20Sensores.pdf
Consulta Febrero 2015.

[6] CODISA Ingenieros, *La importancia de tener un buen sistema de monitorización*.
http://www.codisaingenieros.es/noticias/11-redes/52-la-importancia-de-tener-un-buen-sistema-de-monitorizacion
Consulta Febrero 2015.

[7] Wikipedia, *Sistemas de monitorización y control*.
http://es.wikipedia.org/wiki/Sistemas_de_monitorizaci%C3%B3n_y_control
Consulta Febrero 2015.

[8] NC State University, ECE Department, *Embedded Computer Systems*.
http://www.ece.ncsu.edu/research/cas/ecs
Consulta Febrero 2015.

[9] Wikipedia, *Sistema embebido*.
http://es.wikipedia.org/wiki/Sistema_embebido
Consulta Febrero 2015.

[10] GINGA, *ginga.org*.
`http://www.ginga.org.br/es`
Consulta Febrero 2015.

[11] Rhode&Schuarz, *Rhode&Schuarz*.
`http://www.rohde-schwarz.com/`
Consulta Febrero 2015.

[12] Rhode&Schuarz, *ETL TV Analyser*.
`http://www.rohde-schwarz.com/en/product/etl-productstartpage_63493-9255.html`
Consulta Febrero 2015.

[13] xRefs, *Codigo para soporte para chips dibCom en kernel de Linux*.
`http://www.xrefs.info/linux-3.16/search?q=dibcom&defs=&refs=&path=&hist=&type=`
Consulta Febrero 2015.

[14] Geniatech, *S870*.
`http://www.geniatech.com/pa/s870.asp`
Consulta: Febrero 2015.

[15] Jose Joskowicz, Rafael Sotelo, J. Carlos Lopez Arado, *Comparison of Parametric Models for Video Quality Estimation: Towards a General Model*.

[16] Wikipedia, *Mean Opinion Score*.
`http://en.wikipedia.org/wiki/Mean_opinion_score`
Consulta Febrero 2015.

[17] Jose Joskowicz, Rafael Sotelo, J. Carlos Lopez Arado, *Comparison of Parametric Models for Video Quality Estimation: Towards a General Model*.

[18] Video Quality Indicators, *Publicaciones*.
`http://www2.um.edu.uy/ingenieria/vqi/publicaciones.htm`
Consulta Febrero 2015.

[19] Video Quality Indicators, *ProyectoVQI*.
`http://www2.um.edu.uy/ingenieria/vqi/inicio.htm`
Consulta Febrero 2015.

[20] FFMpeg, *Download FFmpeg*.
`https://www.ffmpeg.org/download.html#get-sources`
Consulta Febrero 2015.

[21] Venegas Picon Luis, *BTS Open Caster*.
`http://tesis.pucp.edu.pe/repositorio/bitstream/handle/123456789/1393/VENEGAS_PICON_LUIS_BTS_OPENCASTER.pdf?sequence=1`
Consulta Enero 2015.

[22] Wikipedia, *H.264*.
`http://en.wikipedia.org/wiki/H.264`
Consulta: Julio 2014.

[23] Iain E. Richardson, *The H.264 Advanced Video Compression Standard, Second Edition*.

[24] Ubuntu Manuals, *mediainfo*.
`http://manpages.ubuntu.com/manpages/precise/man1/mediainfo.1.html`
Consulta: Febrero 2015.

[25] VirtualDub, *VirtualDub*.
`http://www.virtualdub.org/`
Consulta: Febrero 2015.

[26] AviSynth, *AviSynth*.
`http://avisynth.nl/index.php/Main_Page`
Consulta: Febrero 2015.

[27] Jorge Visca, Javier Baliosian, Eduardo Grampín, *A Distributed Notification Bus for Constrained Devices*.

[28] Webfs, *Webfsd*.
`http://linux.bytesex.org/misc/webfs.html`
Consulta: Julio 2014.

[29] die.net, *ftpd(8) - Linux man page*.
`http://linux.die.net/man/8/ftpd`
Consulta: Febrero 2015.

[30] NTP Brasil, *NTP*.
`http://www.ntp.br/`
Consulta: Julio 2014.

[31] EECIS, *NTP Overview*.
`http://www.eecis.udel.edu/~mills/database/brief/overview/overview.pdf`
Consulta: Julio 2014.

[32] Linux Hispano, *Cómo configurar NTP*.
`http://www.linuxhispano.net/2010/11/27/configurar-ntp-hora-en-red-en-nuestro-equipo/`
Consulta: Julio 2014.

[33] Archilinux, *NTPD*.
`https://wiki.archlinux.org/index.php/Network_Time_Protocol_daemon_(Espa%C3%B1ol)`
Consulta: Julio 2014.

[34] IETF, *RFC 793*.
`https://www.ietf.org/rfc/rfc793.txt`
Consulta: Febrero 2015.

[35] IETF, *RFC 958 - NTP*.
`https://www.ietf.org/rfc/rfc958.txt`
Consulta: Julio 2014.

[36] IETF, *RFC 959 - FTP*.
`https://www.ietf.org/rfc/rfc959.txt`
Consulta: Julio 2014.

[37] IETF, *RFC 2616 - HTTP 1.1.*
`https://www.ietf.org/rfc/rfc2616.txt`
Consulta: Febrero 2015.

[38] NTP, *NTP.*
`http://www.ntp.org/`
Consulta: Julio 2014.

[39] GNU, *Wget.*
`http://www.gnu.org/software/wget/`
Consulta: Julio 2014.

[40] MythTV, *Tzap.*
`http://www.mythtv.org/wiki/tzap`
Consulta: Julio 2014.

[41] Crontab, *Crontab Manual.*
`http://crontab.org/`
Consulta: Febrero 2015.

[42] Sourceforge, *DVBSnoop.*
`http://dvbsnoop.sourceforge.net/`
Consulta: Julio 2014.

[43] LinuxTV, *DVB-Apps.*
`http://www.linuxtv.org/wiki/index.php/LinuxTV_dvb-apps`
Consulta: Julio 2014.

[44] LinuxTV, *Scan.*
`http://www.linuxtv.org/wiki/index.php/Scan`
Consulta: Febrero 2015.

[45] Wikipedia, *GNU/Linux.*
`http://es.wikipedia.org/?title=GNU/Linux`
Consulta: Febrero 2015.

[46] Ubuntu, *Ubuntu.*
`http://www.ubuntu.com/`
Consulta: Febrero 2015.

[47] SSH, *Open SSH.*
`http://www.openssh.com/`
Consulta: Febrero 2015.

[48] Perl, *Perl.*
`https://www.perl.org/`
Consulta: Febrero 2015.

[49] Oracle, *JAVA SE.*
`http://www.oracle.com/technetwork/java/javase/overview/index.html`
Consulta: Febrero 2015.

[50] LinuxTV, *ISDB-T USB Devices.*
http://linuxtv.org/wiki/index.php/ISDB-T_USB_Devices
Consulta: Febrero 2015.

[51] Geniatech, *S870.*
http://www.geniatech.com/pa/s870.asp
Consulta: Febrero 2015.

[52] Aliexpress, *Full Seg ISDB-T GENIATECH Mygica S870 Digital TV USB Stick for Brazil-Argentina-Peru-Chile.*
http://www.aliexpress.com/store/product/
Full-Seg-ISDB-T-GENIATECH-Mygica-S870-Digital-TV-USB-Stick-for-Brazil-Argentina-Peru-C
616485_699097668.html
Consulta: Febrero 2015.

[53] Acer Support, *Acer Extensa 5620 User Manual.*
http://global-download.acer.com/GDFiles/Document/User%20Manual/
EX_5620_5620Z_5220_OLM_Spa.zip?acerid=633645646164676705&Step1=
NOTEBOOK&Step2=EXTENSA&Step3=EXTENSA%205620&OS=ALL&LC=es&BC=
ACER&SC=EMEA_11#_ga=1.160788669.103621016.1424479293
Consulta: Febrero 2015.

[54] Raspberry Pi, *Model B.*
http://www.raspberrypi.org/products/model-b/
Consulta: Febrero 2015.

[55] Ceibal, *Centro Ceibal para el Apoyo a la Educación de la Niñez y la Adolescencia.*
http://www.ceibal.edu.uy/art%C3%ADculo/noticias/institucionales/
Centro-Ceibal-para-el-Apoyo-a-la-Educacion-de-la-Ninez-y-la-Adolescencia
Consulta: Febrero 2015.

[56] Ceibal, *Hardware Positivo BGH CL11.*
http://www.ceibal.edu.uy/art%C3%ADculo/noticias/consultas/
hardwarepositivo
Consulta: Febrero 2015.

[57] Wikipedia, *Chrome OS.*
http://es.wikipedia.org/wiki/Chrome_OS
Consulta: Febrero 2015.

[58] Comprar un Android TV, *Comprar un CS918 II RK3288 Android 4.4 TV.*
http://www.comprarandroidtv.es/comprar-cs918-ii-rk3288-android-4-4-tv/
Consulta: Febrero 2015.

[59] Mongo DB, *Mongo DB.*
http://www.mongodb.org/
Consulta: Febrero 2015.

[60] Nmon, *Nmon.*
http://nmon.sourceforge.net/pmwiki.php
Consulta: Febrero 2015.

[61] TS Tools, *ts2es*.
`http://tstools.googlecode.com/git-history/70e9b431a7eed2feb4bfece57af310ce25016070/docs/tools.html#ts2es`
Consulta: Febrero 2015.

[62] Google Code, *Código fuente de la herramienta Tzap*.
`https://code.google.com/p/archos-gen8-dvb/source/browse/trunk/sources/LiveTV/jni/zap/tzap.c?r=73#416`
Consulta: Febrero 2015.

[63] Linux Cross Reference, *Código fuente del módulo dib8000 del kernel de Linux*.
`http://lxr.free-electrons.com/source/drivers/media/dvb-frontends/dib8000.c#L3910`
Consulta: Febrero 2015.

[64] Lua, *The programming language Lua*.
`http://www.lua.org/`
Consulta: Febrero 2015.

[65] Wikipedia, *High Definition Video*.
`http://en.wikipedia.org/wiki/High-definition_video`
Consulta: Marzo 2015.

[66] Wikipedia, *Image Resolution*.
`http://en.wikipedia.org/wiki/Image_resolution`
Consulta: Marzo 2015.

[67] Wikipedia, *Secure Shell*.
`http://es.wikipedia.org/wiki/Secure_Shell`
Consulta: Febrero 2015.

[68] Wikipedia, *Audio de alta definición*.
`http://es.wikipedia.org/wiki/Audio_de_alta_definici%C3%B3n`
Consulta: Marzo 2015.

[69] Wikipedia, *High resolution audio*.
`http://en.wikipedia.org/wiki/High-resolution_audio`
Consulta: Marzo 2015.

[70] Wikipedia, *MPEG-4*.
`http://es.wikipedia.org/wiki/MPEG-4`
Consulta: Marzo 2015.

[71] The Interactive Television & Business Index, *The definition of Decoder*.
`http://www.itvdictionary.com/definitions/decoder_definition.html`
Consulta: Marzo 2015.

[72] Wikipedia, *BML*.
`http://en.wikipedia.org/wiki/Broadcast_Markup_Language`
Consulta: Marzo 2015.

[73] TeleMídia Laboratory, *An Introduction to DTV and to Ginga-NCL.*
`http://www.telemidia.puc-rio.br/sites/telemidia.puc-rio.br/files/Introduction%20to%20DTV%20and%20to%20Ginga-NCL.pdf`
Consulta: Marzo 2015.

[74] TeleMídia Laboratory, *Pontifical Catholic University of Rio de Janeiro.*
`http://www.telemidia.puc-rio.br/`

.

[75] Wikipedia, *One-Seg.*
`http://es.wikipedia.org/wiki/One_seg`
Consulta: Marzo 2015.

[76] Wikipedia, *Middleware.*
`http://es.wikipedia.org/wiki/Middleware`
Consulta: Marzo 2015.

[77] Wikipedia, *Advance Audio Coding.*
`http://es.wikipedia.org/wiki/Advanced_Audio_Coding`
Consulta: Marzo 2015.

[78] Wikipedia, *MPEG-2.*
`http://es.wikipedia.org/wiki/MPEG-2`
Consulta: Marzo 2015.

[79] Wikipedia, *ARIB.*
`http://en.wikipedia.org/wiki/Association_of_Radio_Industries_and_Businesses`
Consulta: Abril 2015.

[80] Leonardo Steinfeld, *Despliegue de redes de sensores inalámbricos en la agricutura para mejorar la rentabilidad y la sustentabilidad medioambiental.*
`http://www.universidad.edu.uy/retema/files/2014/09/LSteifneld-RETEMA-2014.pdf`
Consulta: Abril 2015.

[81] IIE-Fing, *Una Experiencia Piloto de Red de Sensores Inalámbricos para Aplicaciones Agronómicas.*
`http://iie.fing.edu.uy/epim2008/programa/p32.pdf`
Consulta: Abril 2015.

[82] Universidad Pontificia Bolivariana, Medellín, Colombia, *Implementación de Redes Inalámbricas de Sensores WSN (Wireless Sensor Networks) para el Monitoreo Estructural.*
`http://kosmos.upb.edu.co/web/uploads/articulos/%28A%29_Implementacion_de_redes_inalambricas_de_sensores_WSN_para_el_monitoreo_estructural_fNGGw.pdf`
Consulta: Abril 2015.

[83] LATU, *LaboratiorioTecnológico del Uruguay.*
`http://www.latu.org.uy/`
Consulta: Abril 2015.

[84] ANII, *Agencia Nacional de Investigación e Innovación.*
`http://www.anii.org.uy/web/`
Consulta: Abril 2015.

[85] DINATEL, *Dirección Nacional de Telecomunicaciones.*
`http://www.dinatel.gub.uy/`
Consulta: Abril 2015.

[86] Escuela Politécnica del Ejército de Ecuador, *Proyecto Analizador de TS.*
`http://repositorio.espe.edu.ec/handle/21000/6014`
Consulta: Abril 2015.

[87] Wikipedia, *Códec.*
`http://es.wikipedia.org/wiki/C%C3%B3dec`
Consulta: Mayo 2015.

Apéndice A

Apéndice

A.1 FTP vs HTTP

Para decidir qué protocolo usar para la transferencia de archivos, se decidió comparar la performance de estos dos protocolos. Los resultados de las pruebas fueron los siguientes. Para analizar la carga que estos protocolos añaden al sistema, como factor crítico se tuvo en cuenta el uso de procesador ya que el uso de memoria resultó ser muy bajo.

Prueba 1

Cantidad de transferencias: 1

Tamaño de archivo: 232 MB

	FTP	HTTP
Uso CPU promedio	2.32%	2.48%
Uso MEM promedio	0.1%	0.02%

Prueba 2

Cantidad de transferencias: 3

Tamaño de archivo: 232 MB / 96 MB / 22 MB

	FTP	HTTP
Uso CPU promedio	3.83%	4.27%
Uso MEM promedio	0.1%	0.02%